VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ADAPTIVE MODEL FOR SIMULATION OF ATMOSPHERIC POLLUTION

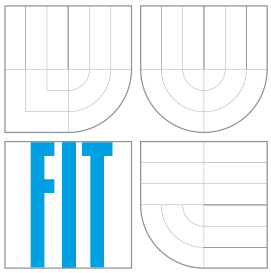
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JANA PAZÚRIKOVÁ

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ADAPTIVNÍ MODEL PRO SIMULACI ZNEČIŠTĚNÍ OVZDUŠÍ

ADAPTIVE MODEL FOR SIMULATION OF ATMOSPHERIC POLLUTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JANA PAZÚRIKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADIM DVOŘÁK

BRNO 2012

Abstrakt

Znečištění ovzduší ohrožuje životní prostředí i životy lidí a je třeba lépe pochopit procesy, které se za ním skrývají. Počítačové modely a jejich simulace pomocí advekčně-difuzní rovnice, popřípade jinými způsoby, poměrně přesně reprezentují pohyb a proměny kontaminantu. Současné modely jsou však validní pouze za určitých omezených počátečních podmínek. V práci je představen obecný model kombinující několik specifických modelů, které jsou schopny měnit se dle vstupních parametrů a zlepšovat se trénováním. Adaptivnost systému je zajištěna rozhodovacím stromem a genetickým algoritmem. Rozhodovací strom reprezentuje datovou strukturu s informacemi pro proces výběru a kombinaci modelů, genetický algoritmus slouží jako optimalizační metoda pro přizpůsobení stromu trénovacími daty. Ohodnocení implementovaného systému dokazuje, že kombinace modelů dává lepší výsledky než modely samotné. I s jednoduchými specifickými modely má systém výsledky srovnatelné se současnými modely znečištění ovzduší.

Abstract

Air pollution harms the environment and human welfare. Computer models and their simulation are useful tools for deeper understanding of processes behind as they quite accurately represent the dispersion and transformation of pollutants with advection diffusion equation or by other concepts. Current models give valid results only to constrained cases of initial conditions. The general model combining the several specific models which is able to change according to input parametres and improve with training is proposed. The adaptiveness of the system is provided by decision tree as data structure with information for selection and combination process and genetic algorithm as optimization method for adjusting the tree. The evaluation of implemented system proves that the combination of models gives better results than models themselves. Even with simple specific models, the system has achieved results comparable to state-of-art models of air pollution.

Klíčová slova

znečištění ovzduší, modelování, simulace, advekčně-difuzní rovnice, rozhodovací strom, genetický algoritmus, umělá inteligence

Keywords

air pollution, modelling and simulation, ADE, decision tree, genetic algorithm, artificial intelligence

Citace

Jana Pazúriková: Adaptive Model for Simulation of Atmospheric Pollution, diplomová práce, Brno, FIT VUT v Brně, 2012

Adaptive Model for Simulation of Atmospheric Pollution

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením Ing. Dvořáka. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

. . . . . .
Jana Pazúriková
18. května 2012

Contents

Chapter 1

Introduction

Air pollution harms the environment and human welfare. In order to solve existing and prevent future problems, the deep understanding of air pollution's processes is necessary. Computer models use various mathematical and statistical concepts to find the relation between input parameters such as source properties, weather conditions and pollutant characteristics, and measured concetrations of the chemical after some time in space. The simulation can provide the complex view, but current models are accurate only for situations they were designed for. The system which would use multiple models and select the most appropriate one for given initial conditions could offer more general perspective. The choosing and combining these specific systems are two non-trivial tasks and due to desired adaptive behaviour of the system artificial intelligence methods might succeed.

Proposed solution uses two of them - genetic algorithm and decision tree. Decision tree contains information about which models to select according to input characteristics and how to combine them into one general model which calculates concentrations. Genetic algorithm adjusts this decision tree by changing the information included and adding new one so that it fits best to training data.

The system has been implemented and extensively evaluated on data from Copenhagen and Cabauw experiment. The results show that the system combining models has not only wider application area but the concentrations calculated by the system are significantly closer to real-world measurements than the concentrations calculated by specific models themselves.

The thesis consists of six chapters. Chapter 2 analyzes the problem and describes air pollution, its modelling and simulation and gives short overview of artificial intelligence methods. Chapter 3 introduces proposed solution, explain its design and main features. In chaper 4, the details of implementation are given. Chapter 5 include evaluation of the system. Finally, chapter 6 states the conclusions.

Chapter 2

Problem Analysis

Problem with air pollution dates back to middle ages and it causes large number of deaths and respiratory diseases every year [EHSZ08]. In order to prevent and improve these issues, we need to better understand the main principles of air pollution. Computer models can provide insight to the concept, inspect contributing factors, assess their importance and simulate various case studies. Although current models represent quite accurately the process of polluting with initial conditions they were designed to illustrate, the widely applicable system combining two or more specific models according to input parameters is missing. Such system would provide more general results and give the overall perspective. However, combining these models is a non-trivial task and choosing the right models to combine is similarly difficult to do. The adaptiveness of the system may be achieved by methods of artificial intelligence.

Background in three main fields is necessary for analysis of the problem - air pollution, modelling and simulation and artificial intelligence methods. First, the processes behind air pollution are described and main mathematical models presented. They usually have form of differential equations so the overview of approaches for solving these equations by computer is given. The examined approach of using artificial intelligence methods for combining different models require introduction to these methods.

2.1 Air Pollution Modelling

Air pollution is

> the presence of contaminants or pollutant substances in the air that interfere with human health or welfare, or produce other harmful environmental effects [Val08].

By its modelling we can describe the functional relation between emissions of pollutants and measured concentrations. Moreover, it gives thorough look into the process, quantify the influence of the parameters, analyzes the possible consequences of case scenarios and assess how effectively proposed strategies reduce the risk. [Bui01].

In these simulations, we try to monitor the fate of pollutants. They undergo two types of processes - physical transport and chemical transformation. The main factors which have influence on them, therefore the input information to the system, are emission's source, meteorological characteristics, terrain, nature of pollutant and atmospheric emissions.

2.1.1 Factors of Air Pollution

Emission's source Source contributes to the model with two aspects - the physical properties of the source itself and the characteristics of the emission process. First part includes source's position, elevation, physical height of the stack and shape. Second part includes emission rate (usually in gs^{-1}), momentum of released pollutants, duration and character of emission (instantaneous, continuous, puff).

Meteorological characteristics Climate and weather at the time and place of emission are the most important influencers of the particles's movement. They include wind direction and speed (usually as functions of height, as the wind speed tends to increase and direction to shift with increasing elevation), temperature and pressure of the air, and turbulence characteristics. Wind velocity influences the transport of contaminants, especially for point sources is this one parameter crucial.

The wind is not blowing straight in one direction, it swirls in irregular, seemingly random way. This movement called turbulence is caused by moving past objects (mechanical turbulence) or, usually, by rising of heated air from the surface and descent of surrounding air (thermal turbulence). Turbulence is responsible for dispersion of the contaminants. In the process of eddy diffusion the atmospheric eddies break apart and mix unpolluted air with polluted air. To estimate the level of turbulence and its dispersive ability, Pasquill classification of atmospheric stability based on wind speed, insolation and cloudiness [Val08, p.562] categorizes the turbulence into six classes from A(unstable) to F(stable), see table 2.1. Generally, during the day sun heats the bottom air which starts to rise up and cause instability. At night the bottom air cools down and atmosphere is stabilized [SJ05]. Pasquill-Gilford tables provide vertical and horizontal dispersion parameters for diffusion equation. Turbulence occurs only up to a certain height above the ground, the depth of this atmospheric layer is called mixing height or height of atmospheric/planetary boundary layer. The air above is stable.

Table 2.1: Pasquill Stability Categories [Val08, p.562].

Surface wind speed (ms^{-1})	Insolation			Night	
	Strong	Moderate	Slight	Thinly overcast or $\geq$ 4/8 low cloud	$\leq$ 3/8 cloud
<2	A	A-B	B	-	-
2-3	A-B	B	C	E	F
3-5	B	B-C	C	D	E
5-6	C	C-D	D	D	D
>6	C	D	D	D	D

Terrain Roughness of the surface and obstacles in the way can alter the wind direction or cause mechanical turbulence. Therefore, a different model is required for the city center than for corn fields. For example, streets with high buildings change the wind properties so much that they form wind canyons.

Temporal and Spatial Scale When examining the pollution process, the time and area scale need to be determined. They are interconnected, because it takes time for a pollutant to get further or change. Local scale focuses on nearby zone, up to 5 km. In this area in short time large particles and fluorides settle down in concentration gradient. In urban scale, up to 50 km, first chemical transformations take place, e.g. nitrogens and sulfurs begin to oxidate. They can cause acid rain in regional scale up to 500 km. Continental (up to several thousand km) and global scale deal with greenhouse gases and ozone holes.

Pollutant Pollutant itself influences the process. First, its physical phase and size determines how long it can stay in the air. The very fine particulate matter and non-reactant gases can remain in the atmosphere for very long time. Otherwise, the particles undergo one of the following processes. Sedimentation or settling by gravity needs to be considered in case of large particles (more than $20\mu m$) in e.g. tilted plume model. Smaller particles are either transformed into another chemical by reaction which is usually modelled as an exponential decay with time. The rest is removed by deposition. Dry deposition is impaction with vegetation or chemical reactions with ground, wet deposition take the pollutants out from the atmosphere by rainout or washout.

The substances directly emitted into atmosphere (primary pollutants) from sources include [ZD07]

- carbon compounds, e.g. CO, CO_2, CH_4,
- nitrogen compounds, e.g. NO, N_2O, NH_3,
- sulfur compounds, e.g. H_2S, SO_2,
- halogen compounds, e.g. chlorides, fluorides, bromides,
- particulate matter.

These primary pollutants form secondary pollutants by chemical reactions. They include

- NO_2 and HNO_3 from NO,
- O_3 from photochemical reactions of nitrogen oxides,
- acid droplets from SO_2 and NO_2,
- sulfates and nitrates aerosols,
- organic aerosols.

EPA (US Environmental Protection Agency) defined air criteria pollutants which need to be monitored as particulate matter, O_3, CO, SO_2, NO_2 and Pb. [Val08, p.58].

The density and temperature of pollutant also play a role because they determine the buoyancy, the tendency to rise up, which sets the actual, effective height of emission (in comparison to the physical height of stack). Gases of higher density than the surrounding air go downwash, others rise up more or less and are exposed to turbulence within mixing height.

Atmospheric emissions For most accurate modelling of chemical transformation the model has to include also the emissions already present in the air.

Air dispersion models (transport and diffusion) estimate the concentrations of pollutant at the specific locations after given period of time. They compute an expected mean of concentration, but they can not predict specific realization [HPM08]. The source's properties, meteorological conditions and terrain characteristics of the area are parameters. Mostly used mathematical concepts are Gaussian model, Lagrangian model, box model, Eulerian model, dense gas model and few others. Another way to simulate air pollution is to find relationship between available data by statistical methods, neural networks or fuzzy logic. Air quality models (chemistry) simulate the chemical reactions that can occur if released primary pollutants are in the atmosphere for longer time and they form secondary pollutants. These transformations include simple reaction, oxidations, and photo/thermochemical chain reactions. Receptor models try to solve the reversed problem, to identify and connect the known pollutants's concentrations to their probable sources.

Applications of air pollution modelling extend especially to environment issues. Simulations are able to assess the consequences of accidental leakage of contaminant in a factory or terrorist attack. Also, the effectiveness of control and emergency mechanisms can be examined and importance of specific influences on process calculated. Therefore, modelling and simulation of air pollution offer much more than mere concentrations measurements.

2.1.2 Advection-Diffusion Equation

Advection-diffusion equation (ADE) [SJ05] is the central model for air pollution modelling as it is valid on all spatial scales. It can be derived from mass balances of transported particles. Symbols used in the equation are explained in appendix D and here:

$C(\mathbf{x}, t)$ concentration at space in time,

t time,

$\mathbf{x} = (x, y, z)$ position in coordinate system, x downwind distance, y crosswind distance, z receptor height from ground,

$u(\mathbf{x}, t) = (u, v, w)$ wind velocity,

$E(\mathbf{x}, t) = (E_x, E_y, E_z)$ diffusion coefficient,

Advection equation describes transport of pollutant in the flow of wind with velocity $\mathbf{u}$.

$$\frac{\partial}{\partial t} C(\mathbf{x}, t) + \frac{\partial}{\partial x} (\mathbf{u}(\mathbf{x}, t) C(\mathbf{x}, t)) = 0 \tag{2.1}$$

Diffusion equation describes diffusion of pollutant due to turbulence with coefficient $\mathbf{E}$. The original diffusion equation describes the random molecular movements in liquids which can be neglected if particles move. However, same equation, only with higher coefficient, governs also turbulent motions [SJ05].

$$\frac{\partial}{\partial t} C(\mathbf{x}, t) = \frac{\partial}{\partial x} (E(\mathbf{x}, t) C(\mathbf{x}, t)) \tag{2.2}$$

Advection-diffusion reaction combines equations 2.1 and 2.2.

$$\frac{\partial}{\partial t} C(\mathbf{x}, t) + \frac{\partial}{\partial x} (\mathbf{u}(\mathbf{x}, t) C(\mathbf{x}, t)) = \frac{\partial}{\partial x} (E(\mathbf{x}, t) C(\mathbf{x}, t)) \tag{2.3}$$

The advection-diffusion equation derivation is based on superposition of advection and diffusion. The additive combination is valid as they are independent - the random movement of diffusion does not depend on flow motion. Some known solutions are given in section 2.2.5.

2.1.3 Gaussian Model

Gaussian plume model is one of the known solutions to the diffusion equation wherein wind and diffusion parameters remain constant. It is used for predicting the concentrations from a continuously emitting point of buoyant air pollution such as industrial stacks [EHSZ08]. The equation sets three dimensional system - downwind (x axis), crosswind (y axis) and vertical originating from ground (z axis). Model assumes that concentrations are proportional to the emission rate, the length of their trajectory is proportional to the wind speed and that time-averaged concentrations can be described by Gaussian distributions with standard deviations empirically related to the levels of turbulence. Usually, Pasquill classification of atmospheric stability (see table 2.1) is used and according to meteorological conditions, one of the stability classes is chosen. Pasquill-Gifford tables then offer empirically assessed dispersion parameters in terms of distance from source. The equations are given in section 4.4.

2.1.4 Lagrangian and Eulerian Model

In Lagrangian model, motion of air pollution plumes's particles is modelled as random walk process with moving frame reference. Model assumes that particles remain unchanged the whole time (therefore used for non-reactant gases, such as SO_2). Statistic methods on trajectories of these particles calculate the dispersion. It is used for modelling long time periods. Eulerian model is similar, it tracks the motion of large number of particles in the fixed frame reference of 3D Cartesian grid.

2.1.5 Transformation of Pollutants

Transformation of pollutants due to radioactive decay or chemical reaction can be added to ADE. First, definition of transformation - reaction equation.

$$\frac{\partial}{\partial t}C(x,t) = f(C(x,t),x,t) \tag{2.4}$$

Advection-diffusion-reaction equation (ADRE) combines all three influencers of the pollutant particles.

$$\begin{aligned}&\frac{\partial}{\partial t}C(x,t) + \frac{\partial}{\partial x}(u(x,t)C(x,t)) = \\ &\frac{\partial}{\partial x}(E(x,t)C(x,t)) + f(C(x,t),x,t)\end{aligned} \tag{2.5}$$

ADRE models all possible processes which cause the transformation and spread of emitted pollutants over arbitrary time and spatial scale.

2.2 Modelling and Simulation

A model simplifies the real system (interconnected set of components) in significant properties so it represents system accurately enough. By changing parameters of modelled system

and observing the consequent changes, i.e. by simulating, it is possible to examine the system repeatedly and predict its future behaviour. Models of air pollution are mostly mathematical, statistical or deterministic. Statistical models are based on statistical relations between empirically measured emissions and concentrations. Models which apply neural networks or fuzzy logic to find the relation are considered to belong to this group [Bui01]. Deterministic models use mathematical equations to describe the pollution - partial differential equations for transport and diffusion and ordinary differential equations for chemistry processes [Jac05, ch.6].

These partial differential equations, ADE and ADRE, represent conservation laws for the mass, energy and momentum. As it is not possible to calculate the solution to differential equations straightly by computer, two main approaches exist. First, couple of analytical solutions to differential equations are known. However, they are valid only under specific conditions. Second approach is to numerically approximate the solution by discretization. The calculation breaks equation to three equations - first it computes change caused by diffusion, then transport and then chemistry. This cyclic method called operator splitting alternates the order of three partial equations. The solutions to these continuous differential equations are approximated by algebraic equations in finite number of discrete spatial or temporal nodes. Discretization process consists of three main steps [Kuz10, lecture 1]

- **mesh generation** decomposes space into cells or elements,
- **space discretization** approximates spatial derivatives by finite differences/volumes/elements method,
- **time discretization** approximates temporal derivatives by time steps and schemes such as Euler or Runge-Kutta.

Methods for space approximations are described as follows.

2.2.1 Finite-difference approximation

Finite-difference approximation replace differential operator with a discrete difference analog - spatially grid cells and temporally time steps. It is based on definition of derivative [Isk10]

$$u'(x) = \lim_{\Delta x \to 0} \frac{u(x + \Delta x) - u(x)}{\Delta x} \tag{2.6}$$

Nodal derivatives of the continuous function are reduced to values in finite set of discrete points, spaced by constant Δx. The simplest approximation, known as forward Euler is [Isk10]

$$u'(x_i) \approx \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x} = \frac{u(x_{i+1}) - u(x_i)}{\Delta x} \tag{2.7}$$

The solution of this numerical method approximates the exact solution and the error can be calculated as follows. Taylor series expansion formulate the exact solution in point $x_i + \Delta x$ with known value in x_i by

$$u(x_i + \Delta x) = u(x_i) + \Delta x_i \frac{\partial u}{\partial x}|_{x_i} + \frac{\Delta x^2}{2!} \frac{\partial^2 u}{\partial x^2}\bigg|_{x_i} + \frac{\Delta x^3}{3!} \frac{\partial^3 u}{\partial x^3}\bigg|_{x_i} + \dots \tag{2.8}$$

By rearranging the expression, we get

$$\frac{u(x_i + \Delta x_i) - u(x_i)}{\Delta x_i} - \frac{\partial u}{\partial x}\bigg|_{x_i} = \frac{\Delta x^2}{2!} \frac{\partial^2 u}{\partial x^2}\bigg|_{x_i} + \frac{\Delta x^3}{3!} \frac{\partial^3 u}{\partial x^3}\bigg|_{x_i} + \dots \tag{2.9}$$

So the difference between approximated solution and exact solution of forward Euler method, called truncation error, is sum of the neglected higher-order polynomials of Taylor series.

Another time stepping schemes include Crank-Nicholson scheme, leapfrog scheme, Matsuno scheme, Heun scheme, Adams-Bashforth scheme and widely used Runge-Kutta scheme [Jac05]. Runge-Kutta of fourth order is explicit method which uses weighed average of four points inside the Δx.

Numerical approximation of solution to differential equations replicate the exact solution under several conditions. First, discrete difference has to converge to differential expression. Second, difference has to be consistent which means that truncation error has to approach zero as Δx approaches zero. And third, it has to be stable, the absolute value difference between numerical and exact solution does not grow over time so step truncation errors do not add up. These conditions assure the intuitive behaviour of the correct numerical method that the solution is more exact with decreasing Δx, decreasing time step and increasing order of approximation.

2.2.2 Finite volume approximation

Finite volumes method divides space into discrete volumes surrounding each node and it approximates integral of differential equation over this volume. Moreover, the values in the node are exchanged in flux-conserving manner (the flux entering the volume is same than flux leaving to neighbouring volume). The method can be applied over arbitrary mesh.

2.2.3 Finite element approximation

Finite element method divides space into elements with nodes on their boundaries. Differential equations are approximated by set of simultaneous algebraic equations which represent the behaviour in the element. It would be impossible to find such algebraic equations to describe whole system, but they can quite accurately estimate the small area of element. The mesh is often irregular, refined in areas of interest, large elements on boundaries.

2.2.4 Method of Finite Lines

Method of finite lines solve a partial differential equation by discretizing all but one its dimension. Usually, all spatial dimensions are discretized and time is left to be continuous. That leads to the set of ordinary differential equations which can be solved by numerical integration (e.g. with Runge-Kutta method applied to remaining continuous dimension). It is widely used to solve diffusion or heat equation.

2.2.5 Analytical Solutions

The discretization methods are one approach to approximate the solution of ADE. Another one is to apply known analytical solutions which were devised for some constrained cases. The solution is calculated with given equation and usually only round-off error is present. In some cases, when the approximation takes place in derivation of analytical equation, the solution is only approximated.

Instantaneous point source

Instantaneous point source is the simplest model possible with emission of mass M in $t = 0$ at point $\mathbf{x} = (x_1, y_1, z_1)$. With unchanging one-directional advection with velocity

$\mathbf{u} = (u_x, 0, 0)$ and diffusion $\mathbf{E} = (E_z, E_y, E_z)$, the solution is [SJ05, ap. 2]

$$C(x,y,z,t) = \frac{M}{4\pi t\sqrt{4\pi E_x E_y E_z t}} exp\left(-\frac{((x-x_1)-u_x t)^2}{4E_x t} - \frac{(y-y_1)^2}{4E_y t} - \frac{(z-z_1)^2}{4E_z t}\right) \tag{2.10}$$

With zero velocity and isotropic diffusion $E = E_x = E_y = E_z$, it simplifies to

$$C(x,y,z,t) = \frac{M}{(4\pi E t)^{3/2}} exp\left(-\frac{-r^2}{4Et}\right) \tag{2.11}$$

where $r = \sqrt{x^2 + y^2 + z^2}$ and $x_1 = y_1 = z_1 = 0$.

Instantaneous line source of finite length

For line source of emitted mass m' per length unit in time $t = 0$ of position $\mathbf{x} = (0, 0, \pm z_2)$ with velocity $\mathbf{u} = (u_x, 0, 0)$, the solution is [SJ05, ap. 2]

$$C(x,y,z,t) = \frac{m'}{8\pi t\sqrt{E_x E_y}} \left(erf\left(\frac{z+z_2}{\sqrt{4E_z t}}\right) - erf\left(\frac{z-z2}{\sqrt{4E_z t}}\right)\right) exp\left(-\frac{(x-u_x t)^2}{4E_x t} - \frac{y^2}{4E_y t}\right) \tag{2.12}$$

Continuous point source

With point source emitting from $t = 0$ till now, time t, the solution is [SJ05]

$$\begin{gathered} C(x,y,z,t) = \frac{\gamma\sqrt{\pi}}{2\sqrt{\alpha}} \left(exp(2\sqrt{\alpha\beta})erf\left(\sqrt{\frac{\alpha}{t}} + \sqrt{\beta t}\right) + exp\left(-2\sqrt{\alpha\beta}\right) erf\left(\sqrt{\frac{\alpha}{t}} - \sqrt{\beta t}\right)\right) \\ \alpha = \frac{(x-x_1)^2}{4E_x} + \frac{(y-y_1)^2}{4E_y} + \frac{(z-z_1)^2}{4E_z} \\ \beta = \frac{u_x^2}{4E_x} + k\gamma = \frac{Q\exp\left(\frac{(x-x_1)u_x}{2E_x}\right)}{4\pi\sqrt{4\pi E_x E_y E_z}} \end{gathered} \tag{2.13}$$

Two-dimensional non-stationary ADE

In case of continuous elevated point source, considering only x and z directions of wind and turbulence, unstable turbulence and usual boundary conditions, the solution was derived with generalised integral Laplace transform technique (GILTT) [MVBT06]

$$C(x,z,t) = \sum_{k=1}^{M} \frac{P_k}{t} A_k \overline{C}(x,z,\frac{P_k}{t}) \tag{2.14}$$

where P_k and A_k are roots and weights of the Gaussian quadrature scheme and $\overline{C}$ is Laplace transform technique of the concentration in time.

The GILTT technique for derivation of analytical solution was used also in [WVMB05] and [MVT+05]. The first one enable arbitrary function for eddy diffusivity and the second one utilize eddy diffusivity as function of distance from source.

Other Analytical Solutions

Appendix 2 of [SJ05] gives the thorough list of solutions to mainly theoretical constraints, e.g. for infinite plane source or infinitely continuous emission. Other practical solutions are redundant to specify here due to their extensive definitions. In [LH97], the solution for two-dimensional ADE of non-reactive continuous line source in steady state, with height-dependent wind speed and eddy diffusivities and Robin-type boundary function is given. Some articles focus on finding the equation for turbulent parameters which fits best given other conditions, e.g. [Ulk00, GMCT04, EE07, CSR09].

2.3 Artificial Intelligence Methods

The main feature of general system combining few specific models of air pollution is its adaptiveness. That would be difficult to achieve through rigorous methods, but artificial intelligence approach can provide the ability to change according to input parameters. The method proposed in chapter 3 uses a decision tree and a genetic algorithm, its brief description follows.

2.3.1 Decision Trees

Decision tree is a directed tree used in decision making and in machine learning as a predictive model. The nodes of the tree contain the question or the condition and outgoing edges represent various answers to the question or ways of satisfying the condition. The leaves of the tree hold the final decisions or classes. Most common application of decision trees is in data mining, as the classification method. The trees are formed in the process of supervised learning where they recursively split the data set according to the input variables's values so that all items on the subset have the same value of target variable. After training phase they are able to classify the objects. The decision tree used in the system proposed in this thesis differs from decision trees described, it is not classification method and it is not formed in training process, but it is more a data structure storing information for decision making.

2.3.2 Genetic Algorithms

Genetic algorithm is nature-inspired method which uses codified evolution for optimization of problems. The instance of the problem, represented mainly by variables which are the object of optimization, is somehow coded into computer-understandable format called chromosome. The set of instances forms the population. Each individual of the population undergoes evaluation based on cost (fitness) function. The aim of the method is to minimize (maximize) evaluation value. After evaluation, some individuals are selected from population, usually the best of them with a few inferior ones. The chromosomes of individuals from this subset are modified by mating and mutation operations and new population is built. This cycle repeats itself until the cost (fitness) goes under (above) the given limit or allowed time expires.

2.4 State-of-art of Air Pollution Modelling with Artificial Intelligence Methods

Air pollution models use artificial intelligence methods, but in different way than this thesis proposes. Usually they calculate the relationship between input parameters and measured concentrations, i.e. instead of using ADE, multi-layer perceptron, Kohonen self-organizing net [BM04a, BM04b] or fuzzy logic [DPMH04] find the relation by training data. Also, cellular automata are used for simulation of air dispersion or air quality, e.g. [MRRM00].

More similar system was devised in [HPM08, ch.14] where the genetic algorithm combines the backward looking receptor model with the forward transport and dispersion model to find the possible sources's characteristics from known concentrations. They blend these two models with continuous genetic algorithm as optimization tool for assimilation of chemical mass balance receptor model and Gaussian plume dispersion model. Mathematically, they search the best fitting matrix S to satisfy the equation $C_{mn}.S_n = R_m$. First, they calculate the estimated concentrations with characteristics of multiple possible sources and multiple meteorological conditions by diffusion equation and fill the matrix C_{mn}. The measured concentrations, based on reality, are in matrix R_m. The object which genetic algorithm changes, the matrix S_n describes the contribution of each source to the actual pollution values. The system was tested and evaluated on both artificial and real-world data and „works rather well“ [HPM08, p.294].

2.5 Research Problem and Research Questions

The current research in atmospheric pollution provides several models and analytical solutions, but the general system combining these models is not present. This master's thesis aims to research how the selection and combination of specific systems can be done. How can system make decisions about choice according to input data? How does it set the configuration parameters so that the chosen models fits to the given instance? How can be models combined? We believe that artificial intelligence methods can be applied, the goal is to find out which ones and how exactly. First solution design works with genetic algorithms and variation of decision trees. The hypothesis is that the measured concentrations of pollutants will be similar to the concentrations calculated with our system, given the same input data or, at least, that the results of combined model will be closer to measured data than the results of the specific models themselves.

Apart from this main problem, we would like to examine one more, less important for this research. It is well known that the proper visualization of data can improve the user's understanding and further work with results. The integration of geographic information systems and display of calculated concentrations can make atmospheric pollution models more powerful [EHSZ08]. The corresponding question is how to do it effectively.

The main problem is, there is not a lot of the real-world data and measurements. Genetic algorithm training is based on real-world data so the evaluation of this system might be constrained. Most of the models described in section 2.2.5 are tested and validated on Copenhagen Tracer experiment's measurements [GLRD98], so the test on those data is necessary for comparison.

Chapter 3

Solution Design

Air pollution models capture the process of dispersion and transformation of contaminants accurately but they often work only under certain initial conditions. The general model which is able to adapt to input parameters, select appropriate specific models, combine them into one system and calculate the correct results would provide wide-ranging framework with overall perspective and broader options of use. Proposed adaptive model of atmospheric pollution is to our knowledge first attempt to do so, however, few systems of blending two models together are mentioned in section 2.4. The solution applies decision tree and genetic algorithm to the tasks of combining models and adjusting the system. The decision tree contains information about which models to select according to input characteristics and how to combine them into one general model which calculates concentrations. Genetic algorithm adjusts this decision tree by changing the information in it so that it fits to training data. Further follows the description of the system, its training and producing results.

3.1 Overview of system's processes

The proposed model works from user's point of view like another air pollution model presented in previous chapter. User inserts input variables. The overview of factors which influence the dispersion and transformation of the pollutant is given in section 2.1.1. The actual list of input parameters in the system depends on the chosen and implemented specific models and their applicability to various cases. Then the system calculates the assessed concentrations in temporal and spatial area of interest. However, the main feature of the model is its adaptiveness, the ability to adjust to conditions and improve with acquired knowledge.

The system therefore works in two modes - user mode and training mode as shown in figure 3.1. In user mode, the user gives the input parameters, system builds the combined model with information from the decision tree, calculates and visualizes the results. Training mode provides the means to change the system. After the input of training data which consists of input variables and measured concentrations, the system's results are obtained as in user mode. The calculated and measured results are compared and genetic algorithm changes the decision tree accordingly.

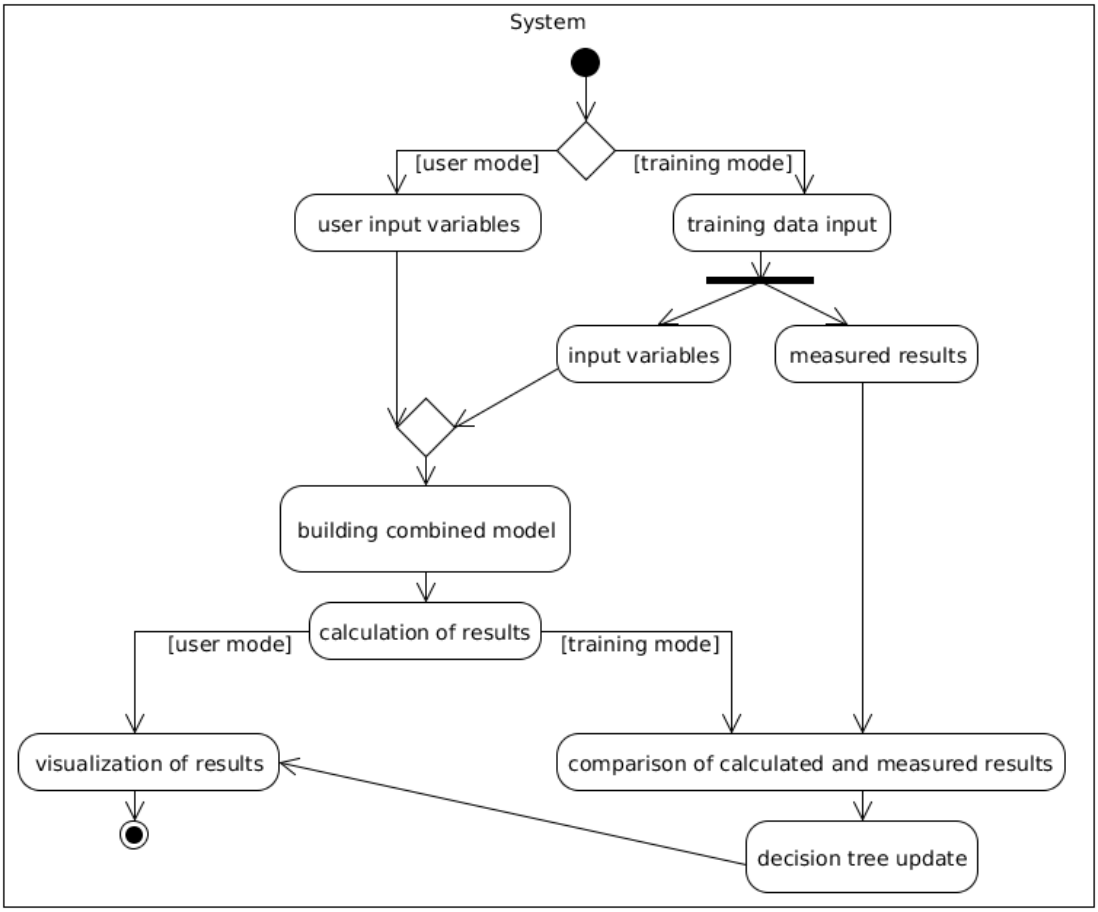


Figure 3.1: Activity diagram of system.

3.2 Decision tree

The decision tree contains all the information about the process of specific models's selection and combination.

The leaves contain implemented specific models with their „accuracy parameter". This parameter, a contribution ratio, is applied in case that two or more specific models are found suitable for the same area. The system then calculates linear combination of their results and coefficients for the equation are determined by the contribution ratios.

Inner nodes represent conditions such as *point source* or *constant wind speed* required for model validity or recommended application to certain input properties, e.g. Lagrangian model is *suitable for long temporal scales* [Bui01, p.8]. The input parameters either do or do not satisfy these conditions, the satisfiability is boolean value, but the suitability of using specific models under these conditions is scalable, therefore has numerical value. The extent to which the conditions need to be satisfied so that the specific model is suitable - „suitability parameter" (value *1.0* - must be/validity requires, *0.7* - should be/it is recommended, *0.5* - might be/does not matter, *0* - can not be/would cause invalidity) are stored in data structure for each implemented model. This offers another way of selecting the most appropriate

model. The threshold value added to inner node can easily divide subtrees to two groups - those with leaves with specific models that has suitability parameter higher than threshold value and those that has lower. However, finding the threshold values so that the system functions effectively, can be a problem. In case of using genetic algorithm described below, that would require extensive training data from many experiments with different input parameters.

Decision process goes from root node and inspects each split condition. The threshold value is compared to boolean value of whether input parameters satisfy split condition in that inner node. If they do, process continues with left subtree, in negative case, it goes right. In case of threshold value 0.5, process goes both directions. Three cases can occur. First, only one model is found as appropriate and used, no combining takes place. Second, some conditions can be met only in part (e.g. threshold value 0.5), in that case several leaves are appropriate and their combination is built. Third, no specific model can be used. The simple example of decision tree is in figure 3.2. The nodes contain conditions about temporal and spatial scale and choose one model according to values of those variables given by the user.

The character of decision tree makes it quite easy to add new specific models into the system. The expert just needs to manually insert it into the decision tree and describe the validity conditions. After rerun of genetic algorithm, new model is part of the system with contribution proportional to its accuracy.

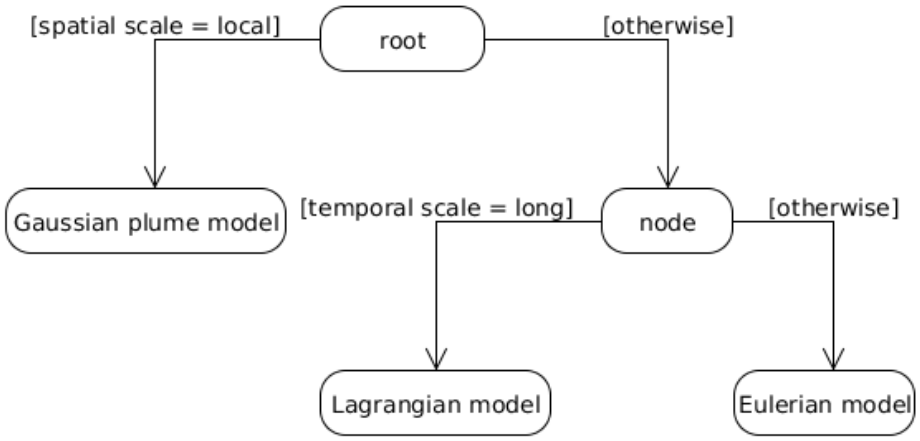


Figure 3.2: Example of decision tree.

3.3 Building combined system

The combining of models into one general system that calculates results (see figure 3.3) expects the decision tree with information necessary for decisions about selection. The building itself is explained by following example. The new industrial stack starts to release continuously non-reactant gas with a few particles of heavy particulate matter. On the right side of the factory, the terrain starts to rise and form a mountain, on the left side the city lies in plane. The user wants to know how the pollution in the area of 15 km will look

in two days if it does not rain. At least two models can be used, first simple tilted plume model for heavy particles, second Lagrangian or Gaussian model for gas. If the system has also model for mountainous terrain it can be included and calculate the results for right side. The sharp transition between left and right side can cause some discrepancies, so the transient area's results can be calculated e.g. as weighted mean of results from two areas. The weights would change with Gaussian distribution dependent on distance from the sharp boundary.

In case of two or more models which were found suitable for same area, the linear combination of their results is calculated as the overall result. The coefficients for combination are determined by contribution ratios of given specific models stored in leaf nodes of tree.

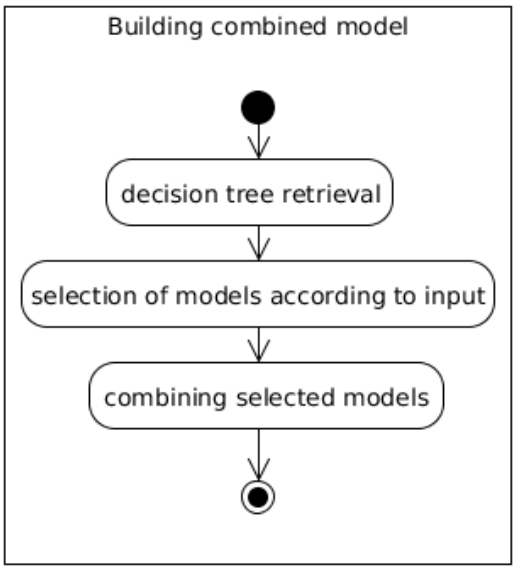


Figure 3.3: Building combined model.

3.4 Adaptation of system

The adaptiveness of the system lies in its ability to change the process of selecting and combining specific models which creates one system calculating the results. This process takes the information from the decision tree, so the data in nodes of the tree is the subject of adjusting so that improved model calculates the results similar (enough) to the measured concentrations which are ground truth in training process. Basically, genetic algorithm optimizes the decision tree so that the combined model built with it calculates the results that fit the measurements.

Genetic algorithm iteratively evaluates the population of individuals represented by their chromosomes and builds new population with operations of mating and mutation from a few chosen (usually the best) individuals, as shown in figure 3.4. Therefore chromosome form, initial population, evaluation process, cost function and methods of selection, mating and mutation need to be determined. The parameters such as population size and maximum generations need to be set experimentally.

Chromozome representation of a tree consists of nodes's list in preorder. For every node, it includes

- id number - the position in preorder,
- split condition,
- split threshold,
- specific model,
- contribution ratio,
- binary flag set if the node can be changed.

For inner nodes, target model and ratio values are set to null. For leaf nodes, split condition and threshold values are set to null. The bit *can be changed* for inner nodes means that threshold values can be changed, for leaf nodes that ratio can be altered. This bit assures that the validity conditions which needs to be satisfied for specific model in the end of path through decision tree, will not be broken by genetic algorithm.

Initial population is built from one individual - initial tree. This tree is created manually by the expert, including all known facts about all chosen implemented specific models. Initial conditions of those models are included in nodes of tree. As these conditions were usually mentioned and proven to be correct in research that presented corresponding model, genetic algorithm tries to find what conditions about other variables can be valid, e.g. with the tree shown in figure 3.2, it might explore the possibility that Lagrangian model works best than others in mountainous terrain in all temporal and spatial scales. Another task is to find contribution ratios for specific models, i.e. how accurate similar models are.

Mutation operations include inserting new node into the tree (right before the leaf nodes), altering the threshold (for newly added nodes) and changing the contribution ratios. Assumably, the contribution ratios for more accurate models is higher, their coefficient in linear combination is higher and therefore they contribute more to the system's results. The need to keep validity conditions in the path from root to the specific model rules out mating and mutation operations over trees as in genetic programming. The path from root to the model should always include the conditions given in the initial tree, but a few more can appear.

Evaluation might take quite long as the decision tree needs to be built for each individual's chromosome from population. Then combined system is built according to each decision tree and it calculates the results with input parameters of training data. The calculated results and results of training data (measured concentrations) are compared and cost function is computed as normalized mean squared error between them. The algorithm ends and the best found decision tree is saved if the error is small enough (or the allowed time expires). Due to probably long evaluation time the population is small, higher number of generations is used to improve the results. Only the best decision tree is selected and new population is built from it by mutation. The parameters of algorithm such as population size, maximum generations or probability of different kinds of mutation needs to be set experimentally so that it converges. The best found tree is then saved and used afterwards in user mode.

As this part of the system is the most important one, the character and parameters need to be tested and experimentally evaluated several times to produce robust and reliable adaptive system.

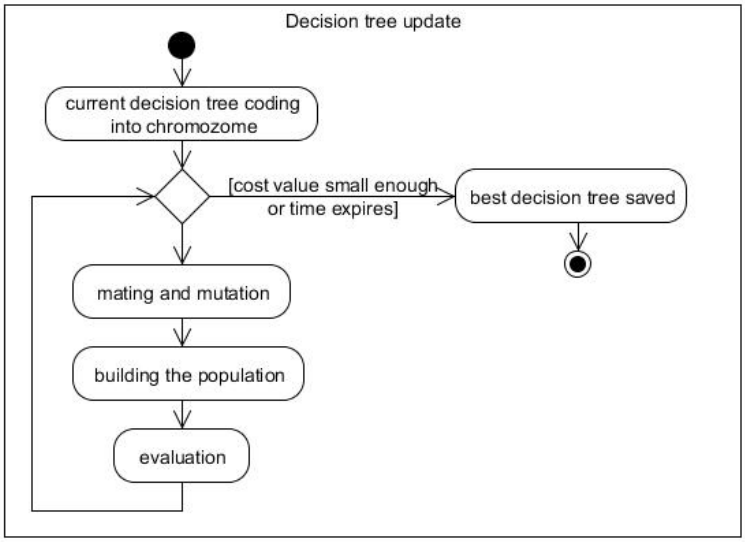


Figure 3.4: Decision tree update.

Designed genetic algorithm takes into account all known facts about specific models, but the changing of decision tree can end in correct and effective results only in case of enough training and evaluation data from whole range of possible input parameters. As there are only a few experiment's data for air pollution and most of them with similar initial conditions, this can pose an issue to evaluation of proposed model.

Chapter 4

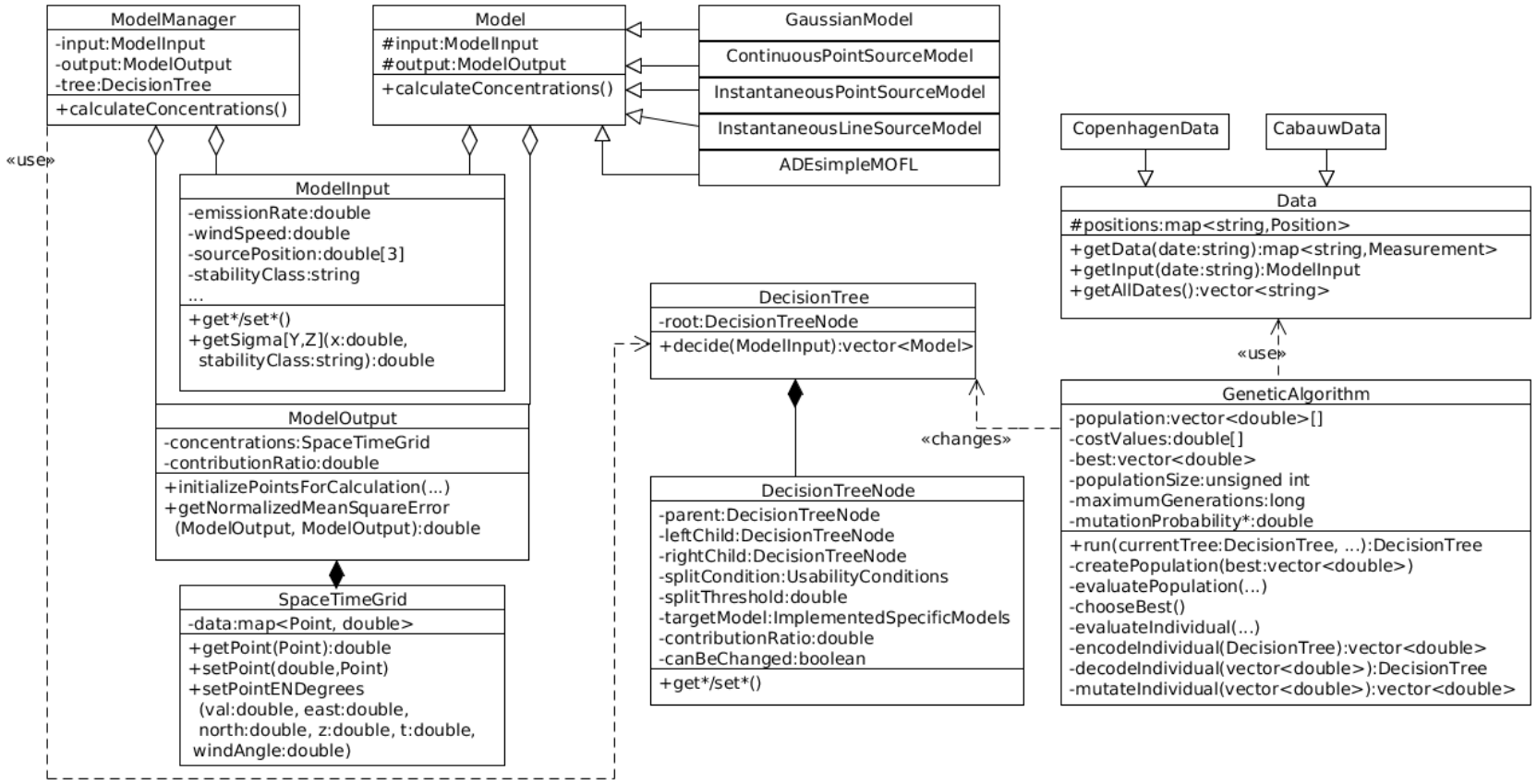
Solution Implementation

The designed system has been implemented in C++ language. From implemented specific models, the implementation of the numerical approximation has been inspired by Radim Dvořák's code [DvZ09]. Apart from STL library, RapidXML [Kal06] has been applied for saving the decision tree in XML format. The random numbers needed in genetic algorithm have been provided by *rand()* function of C language for uniform distribution and by function implemented in C by [Ack02] for normal distribution. Graphical user interface has been designed and implemented in Qt. Apart mentioned cases, all code is originally implemented.

4.1 Overview

Implemented system consists of several classes, as shown in figure 4.1. In user mode, *Model* calculates the *ModelOutput* according to *ModelInput*. *ModelManager* does the same thing, but with the models selected by *DecisionTree*. In training mode, user can compare measured concentrations and results calculated by *Model* or *ModelManager* or *GeneticAlgorithm* changes the *DecisionTree* with training *Data*. The description of each system's functionality follows.

Figure 4.1: Class Diagram of Implemented System.



4.1.1 User Mode

The usual activity flow when user mode is on (the user itself inserts input parameters, selects model and sees the results) goes as follows:

- user sets input parameters,
- user selects the specific model or decision tree (initialized, saved, their own in valid XML format) to be used for calculation of results,
- system constructs *Model* (if one specific model has been selected) or *ModelManager* (if decision tree has been selected),
- system sets given input to the *Model(Manager)* and initializes the points in output where the concentrations will be calculated,
- if one specific *Model* has been selected, it calculates the results,
- if decision tree has been selected, *ModelManager* uses the decision tree and input parameters to select appropriate specific model(s) and then calculates the results with them,
- system visualizes the output.

4.1.2 Training Mode

Training mode offers more interaction with system than the mere calculation of concentrations with given input parameters. It is possible to compare the results calculated with selected model or combined system created according to the decision tree with data from experiments and analyze the error between them. The usual flow goes as follows:

- user selects experiment's data - input parameters and output concentrations,
- user selects the specific model or decision tree (initialized, saved, their own in valid XML format) to be used for calculation of results,
- system constructs *Model* (if one specific model has been selected) or *ModelManager* (if decision tree has been selected),
- system sets input parameters from experiment to the *Model(Manager)* and initializes the points in output according to those where the concentrations were measured in experiment,
- if one specific *Model* has been selected, it calculates the results,
- if decision tree has been selected, *ModelManager* uses the decision tree and input parameters to select appropriate specific model(s) and then calculates the results with them,
- the calculated and measured output are compared, the normalized mean square error is calculated,
- system visualizes both outputs or the difference between them.

Another functionality in training mode is to improve the decision tree with selected real-world data by genetic algorithm, details described in section 4.6.

Table 4.1: Most important *Model Input* member variables representing input parameters.

source's properties	**weather conditions**
source position in 3D Cartesian coordinate system	wind speed
emission rate	wind direction
stack diameter	ambient temperature
exit temperature	atmospheric boundary layer height
exit velocity	stability class
effective height of source	roughness length
length of line source	Monin-Obhkukov length
spatial scale	**pollutant's properties**
maximum distance from source	decay rate
stepX, stepY, stepZ	depositional velocity
receptor height	gravitational velocity

4.2 Model, Model Input and Model Output

Class ***Model*** is the superclass for all implemented specific models. Its main method *calculateConcentrations()* takes input parameters in *ModelInput*, calculates the concentrations and puts the values into *ModelOutput*. List of all implemented specific models is given in section 4.4.

Class ***ModelInput*** includes all input parameters of model. The most important ones (used in evaluation) are listed in table 4.1. Also, the class has the structure containing boolean values whether this input parameters satisfy conditions or not. Apart from getters and setters for these private member variables, methods for calculating σ_y and σ_z using table 4.3 and 4.2 and effective height stack using equation 4.1 are provided.

$$effective\ height\ = h + ((1.6 \exp(\frac{log(f_0)}{3})) * \frac{\exp(\frac{log(3.5x_0)}{3})}{u_x}$$

$$f_0 = 3.12 * 0.785 * exitVelocity * diameter^2 * \frac{exitTemperature_{Kelvin} - ambientTemperature_{Kelvin}}{exitTemperature_{Kelvin}} \tag{4.1}$$

$$x_0 = 34 * \exp(0.4 log(f_0))\ if\ f_0 > 55$$
$$x_0 = 14 * \exp(0.625 log(f_0))\ if\ f_0 <= 55$$

ModelOutput is class representing the concentrations in space and time. Basically, they are stored in *map<Point, double>*, where *Point* is class of the point in 3D Cartesian coordinate system (where x axis is downwind, y axis is crosswind and z is elevation above ground) and time variable, and *double* is the value of pollutant's concentration in given *Point*. The class *SpaceTimeGrid* abstracts the map and provides methods for converting from downwind-crosswind coordinate system to system where x axis is to the east from source, y axis to the north and z axis elevation above ground. The *Model Output* class gives another level of abstraction to the *SpaceTimeGrid* with methods for calculating the errors between two models's outputs and initialization of points where the concentrations will be

calculated. The initialization can be done in multiple ways:

- if the system is in user mode, input parameters *stepX, stepY, stepZ* determine how afar from each other the points for calculation will be,
- if the parameters *step** are not set by user, the points are spread evenly in distance $distanceFromSource/10$,
- if in training mode, points can be initialized according to where measurements of experiment were taken,
- if another *ModelOutput* is already initialized, points can be put according to it.

Helper class ***ModelsConditions*** include the enumerations of names of implemented specific models and all possible split conditions for decision trees. It offers methods and overloaded operators for convenient work.

4.3 Model Manager and Decision Tree

ModelManager is used as an abstraction for *Model* if instead of one, more specific models are supposed to be used for calculation of results. That can happen when decision tree decides which models are appropriate. From user's point of view, it works like when only one model calculates the results. Input parameters are set, output points initialized, concentrations calculated by function *calculateConcentrations()*. The actual method of calculation instead of equations or approximations consists of following steps:

- decision tree decides which specific models are suitable for given input parameters and returns a vector of them with their contribution ratios set in *ModelOutput*,
- for each suitable model, the results are calculated as usual, all with same input parameters and for same points,
- coefficients for linear combination are calculated as $c_i = \frac{contributionRatio_i}{\sum_{i=0}^{n} contributionRatio_i}$ where n is number of suitable models,
- linear combination of specific models's results is calculated and given as the result.

ModelManager has in *map<ModelsConditions::ImplementedSpecificModels, map<ModelsConditions::UsabilityConditions, double> >* all information about validity conditions essential for models and recommended use of models under certain conditions.

DecisionTree is the most important data structure in the system. Its main function *decide* takes model's input and returns vector of suitable models found in decision process in current tree. The function *initializeTree* initializes manually created initial tree of models containing all validity conditions. Tree is saved in XML format to the file „tree.xml“ in working directory and then loaded at application's start.

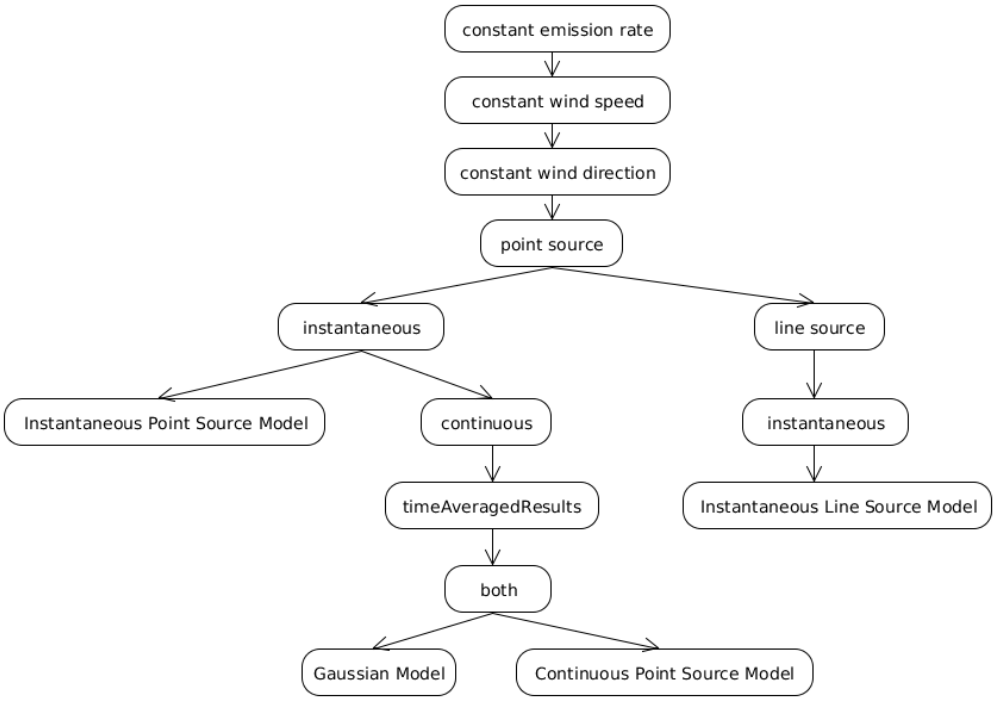
The only member variable is the root node. Decision tree's node has member variables:

- pointer to parent,
- pointers to left and right children,
- split condition as enumeration *UsabilityConditions* member,

- split threshold as *double*,
- target model as enumeration *ImplementedSpecificModels* member,
- contribution ratio as *double*,
- can be changed as *boolean*.

Initial tree has been created manually. In figure 4.2 are shown all meaningful nodes, the missing child nodes are *null* - no appropriate model has been found. The validity conditions found for each specific model (also stated in following section) were arranged as inner nodes, so that they belong to the specific model at the end of the path. Initial contribution ratios are 1, split thresholds 1 as all conditions are „must“. Bit can be changed is set for all leaf nodes and is false for all inner nodes in the initial tree.

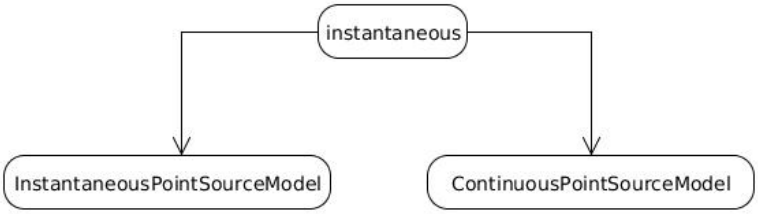
Figure 4.2: Initial, manually created decision tree.



The XML format is easily readable by human, as shown in a following very simple example of tree in figure 4.3. This tree does not include all validity conditions for models, it is only used as an example.

```
<?xml version="1.0" encoding="utf-8"?>
<decisionTree>
  <node id="0" splitCondition="instantaneous" splitThreshold="1"
   canBeChanged="0">
      <node id="1" targetModel="InstantaneousPointSourceModel"
       contributionRatio="1" canBeChanged="1"/>
```

Figure 4.3: Simple tree corresponding to XML example.




```
      <node id="2" targetModel="ContinuousPointSourceModel"
       contributionRatio="1" canBeChanged="1"/>
  </node>
</decisionTree>
```

4.4 Implemented Specific Models

All specific models inherits the member variables *ModelInput* and *ModelOutput* and has to override the method *calculateConcentrations* of the super class *Model*. The analytical model simply solve the equations by replacing the coordinate variables with values in given point, so the calculation is fast even for large number of points. However, numerical solution has to approximate the solution of advection-diffusion-reaction equation for rather dense grid with size of input parameter *distanceFromSource* and after that assign the points in *ModelOutput* their concentration values. So concentrations are calculated not only for points we need (e.g. 500 m from source), but for all points from source to the furthest point. For that reason, numerical approximation takes quite long for distances more than 500 m and it is left out of genetic algorithm's run.

In many analytical solutions, the dispersion parameters are needed instead of diffusion coefficients. The tables 4.2 and 4.3 determine their values by distance from source and Pasquill stability class. All symbols used in equations in this section are explained in appendix D.

Table 4.2: Dispersion Parameters for Urban Area by Briggs [Val08, p.565].

Pasquill stability class	σ_y, m	σ_z, m
A-B	$0.32x(1+0.0004x)^{-0.5}$	$0.24x(1+0.001x)^{0.5}$
C	$0.22x(1+0.0004x)^{-0.5}$	$0.2x$
D	$0.16x(1+0.0004x)^{-0.5}$	$0.14x(1+0.0003x)^{-0.5}$
E-F	$0.11x(1+0.0004x)^{-0.5}$	$0.08x(1+0.0015x)^{-0.5}$

As the implementation of specific models is not the main aim of this thesis, simple, understandable, mostly analytical solutions has been chosen. List of implemented specific models follows, with their validity conditions.

4.4.1 Gaussian Model

Gaussian model gives analytical equations 4.2, 4.3, 4.4 for different conditions [Val08, ch.21, p.558] for calculation of time-averaged concentration in point x, y, z. Solution assumes steady, continuous point source and one-directional constant wind velocity. For stable conditions or very high atmospheric boundary layer

$$C(x,y,z) = Q\frac{1}{u_x} * \left[\left(\frac{g_1}{\sqrt{2\pi}\sigma_y}\right)\right] * \left(\frac{g_2}{\sqrt{2\pi}\sigma_z}\right) \tag{4.2}$$

For unstable or neutral conditions where $\sigma_z > 1.6L$

$$C(x,y,z) = Q\frac{1}{u_x}\frac{q_1}{\sqrt{2\pi}\sigma_y}\frac{1}{L} \tag{4.3}$$

For unstable or neutral conditions, where $\sigma_z < 1.6L$

$$C(x,y,z) = Q\frac{1}{u_x}\frac{g_1}{\sqrt{2\pi}\sigma_y}\frac{g_3}{\sqrt{2\pi}\sigma_z} \tag{4.4}$$

For all equations above, g_1, g_2, g_3 are calculated as follows.

$$\begin{aligned} g_1 &= \exp\left(\frac{-0.5y^2}{\sigma_y^2}\right) \\ g_2 &= \exp\frac{-0.5(h-z)^2}{\sigma_z^2} + \exp\frac{-0.5(h+z)^2}{\sigma_z^2} \\ g_3 &= \sum_{N=-\infty}^{\infty}\left(\exp\left(\frac{-0.5(h-z+2NL)^2}{\sigma_z^2}\right) + \exp\left(\frac{-0.5(h+z+2NL)^2}{\sigma_z^2}\right)\right) \end{aligned} \tag{4.5}$$

According to [Val08], in calculation of g_3, sum from -4 to 4 is sufficient approximation, in the implementation -10 to 10 has been used.

4.4.2 Model for Continuous Point Source

Model for continuous point source gives analytical equation 4.6 [Kin09, ch.5, p.124] for calculation of time-averaged concentration in point x, y, z with source in position x_0, y_0, z_0.

Table 4.3: Dispersion Parameters for Rural Area by Briggs [Val08, p.565].

Pasquill stability class	σ_y, m	σ_z, m
A	$0.22x(1+0.0001x)^{-0.5}$	$0.20x$
B	$0.16x(1+0.0001x)^{-0.5}$	$0.12x$
C	$0.11x(1+0.0001x)^{-0.5}$	$0.08x(1+0.0002x)^{-0.5}$
D	$0.08x(1+0.0001x)^{-0.5}$	$0.06x(1+0.0015x)^{-0.5}$
E	$0.06x(1+0.0001x)^{-0.5}$	$0.03x(1+0.0003x)^{-1}$
F	$0.04x(1+0.0001x)^{-0.5}$	$0.016x(1+0.0003x)^{-1}$

Diffusion coefficients are assessed by equations 4.7 and 4.8, σ_y and σ_z as usual. Solution assumes steady, continuous point source and one-directional constant wind velocity.

$$C(x,y,z) = \frac{Q}{4 * \pi * (x - x_0) * \sqrt{E_y * E_z}} \left(\exp\left(\frac{-(y-y_0)^2 * u_x}{4 * (x - x_0) * E_y} - \frac{(z-z_0)^2 * u_x}{4 * (x - x_0) * E_z} \right) \right) \tag{4.6}$$

$$E_y = \frac{\sigma_y^2 * u_x}{2x} \tag{4.7}$$

$$E_z = \frac{\sigma_z^2 * u_x}{2x} \tag{4.8}$$

4.4.3 Model for Instantaneous Point Source

Model for instantaneous point source gives analytical equation 4.9 [SJ05, ap.2, p.146] for calculation of concentration in point x, y, z in time t after release with source in position x_0, y_0, z_0. Diffusion coefficients are assessed by equations 4.7 and 4.8. Solution assumes steady, point source with instantaneous emission of mass M at time $t = 0$ and one-directional constant wind velocity.

$$\begin{gathered} C(x,y,z,t) = \frac{M}{4\pi t \sqrt{4\pi E_x E_y E_z t}} \\ \exp\left(\frac{-(x - x_0 - u_x)^2}{4E_x t} - \frac{(y-y_0)^2}{4E_y t} - \frac{(z-z_0)^2}{4E_z t} - decayRate * t \right) \end{gathered} \tag{4.9}$$

4.4.4 Model for Instantaneous Line Source

Model for instantaneous line source gives analytical equation 4.10 [SJ05, ap.2, p.147] for calculation of concentration in point x, y, z with source along the line $(0, 0)$ for $z = \pm z_2$. Diffusion coefficients are assessed by equations 4.7 and 4.8. Solution assumes steady, line source with instantaneous emission of mass M per unit length at time $t = 0$ and one-directional constant wind velocity.

$$\begin{gathered} C(x,y,z,t) = \frac{Q}{8\pi t \sqrt{E_x E_y}} * \\ \left(erf \frac{z + z_2}{\sqrt{4 * Ez * t}} - erf \frac{z - z2}{\sqrt{4 * E_z * t}} \right) * \\ \exp\left(\frac{-(x - u_x * t) * (x - u_x * t)}{4 * E_x * t} - \frac{y * y}{4 * E_y * t} - decayRate * t \right) \end{gathered} \tag{4.10}$$

4.4.5 Approximation by Method of Lines

Approximation by method of lines implemented by class *ADEsimpleMOFL* converts real y and z values by Δy and Δz into discrete j and k coordinates, so the equation looks like

$$\begin{gathered} dC(x,j,k) = K_y(x)\frac{C(x,j+1,k) - 2C(x,j-1,k) + C(x,j-1,k)}{\Delta y^2} + \\ K_z(z)\frac{C(x,j,k+1) - 2C(x,j,k) + C(x,j,k-1)}{\Delta z^2} + K\frac{C(x,j,k+1) - C(x,j,k-1)}{2\Delta z} \\ K_y(x) = E_y(x)/u_x \\ K_z(x) = E_z(z)/u_x \\ K = \frac{dE_z}{dz}/u_x \end{gathered} \tag{4.11}$$

.

Diffusion coefficients are calculated and also u_x is approximated by equation by [Ulk00].

$$\begin{gathered} E_y(x) = \frac{\sigma_y^2 u_x}{2x} \\ if \frac{abl}{MoninObhkukovLength} > 0 \\ E_z(z) = vonKarmanConstant * frictionVelocity * z * \frac{1 - \frac{z}{abl}}{1 + \frac{9.2abl}{MoninObhkukovLength}} \end{gathered} \tag{4.12}$$

$$\begin{gathered} else \\ E_z(z) = vonKarmanConstant * frictionVelocity * z * (1 - \frac{z}{abl}) \\ * \sqrt{1 - \frac{13z}{MoninObhkukovLength}} \end{gathered}$$

The initial conditions are

$$\begin{gathered} C(0,0,h) = \frac{Q}{u_z * \delta_z * \delta_y} \\ C(0,y,z) = 0 \ for \ y \neq 0, z \neq h \end{gathered} \tag{4.13}$$

The boundary conditions for borders of 2D mesh (YZ plane) are set to 0 and

$$\begin{gathered} C(x,y,0) = C(x,y,2) \\ C(x,y,h) = C(x,y,h-2) \end{gathered} \tag{4.14}$$

The equation and implementation of numerical method has been inspired by code from supervisor of this thesis. During the method, x coordinates are spaced according to integration step, Δy and Δz set so that the method is stable. Iteratively, for each x, 2D mesh is initialized with y and z axis by Δy and Δz. The Runge-Kutta of fourth order calculates the results and x is increased by adaptive integration step. Concentrations from 2D mesh are put into initialized points in *SpaceTimeGrid* (corresponding value if *SpaceTimeGrid's* point's coordinates match exactly or liner interpolation between two closest points from 2D mesh to the point in *SpaceTimeGrid*). The calculation takes significantly longer than by analytical solution, so numerical approximation is left out from decision tree as training process would take very long.

4.5 Training and Evaluation Data Sets

There is quite a few available data from real-world experiments. Copenhagen experiment is probably the most important one, almost every paper proposing an air pollution model evaluates the results on its data. Furthermore, there are experiments done in Cabauw, Hanford, Nebraska and other locations [Irw12b]. The experiments's input parameters are of low variability - there is always continuous point source emitting non-reactant contaminant and steady state of concentration is measured.

Data sets are represented by abstract class *Data*, superclass for all classes with specific experiment's data. It provides structures for describing the position of receptor and concentration's measurement data. Structure *Position* includes the name of the positions, i.e. its alphanumerical identification, its distance to the east and north from the source. Structure *Measurement* includes x and y coordinates (distance to east and north from source), then up to three concentrations's measurements and the (usually average) concentration value which is used in evaluation. All positions are stored in *map<string, Position>* where *string* is the name of the position. This map is convenient for adding measurement data to the *map<string, Measurement>* (where *string* is name of the position where this measurements was taken).

Data sets have three important methods:

- *map<string, Measurement> getData(string date)* returns measurement's data from the experiment's run on given date,
- *ModelInput* getInput(string date)* return *Model Input* with input parameters of the experiment's run on given date,
- *vector<string>* getAllDates()* return all dates of experiment's runs with usable data (no missing input parameters or measurements) so it is easy to „iterate“ through all runs of experiment.

Data from two experiments done in Copenhagen and Cabauw are put in implementation.

4.5.1 Copenhagen Experiment

Copenhagen experiment [GLRD98] was conducted in Copenhagen urban area during September 1978 - July 1979. The sulphurhexaflouride (SF_6) was released from 115 meters high tower and concentrations after 20, 40 and 60 minutes after release were measured in (usually) 40 receptors's positions up to six kilometers distant from source. Atmospheric conditions were neutral or unstable, meteorological measurements include temperature, wind speed and wind direction in different heights measured every 10 minutes. In the implementation following values were set as input parameters - temperature in 120 m after one hour after release as ambient temperature, average wind speed in 120 m as wind speed in x axis direction and direction of center plume was set as wind direction angle (measured from north counter clockwise). Although data from all runs are put into implemented system, runs on 14.09.1978, 20.09.1978 and 03.11.1978 were omitted in evaluation and testing due to incompletion of their input parameters or concentration measurements, resulting in seven usable experiment runs.

4.5.2 Cabauw Experiment

Cabauw experiment [NAD83] was conducted in Cabauw rural area during April 1977 - October 1978. The sulphurhexaflouride (SF_6) was released from 200m high mast from the height of 80 or 200 meters. Concentrations after 30 and 60 minutes after release were measured in 24 receptors's positions up to four kilometers from source. Atmospheric conditions were mostly stable or neutral and extensive meteorological measurements were taken, including wind speed and direction, temperature, turbulence, radiosonde soundings, acoustic radar and radiation measurements and synoptical observations. In the implementation following values were set as input parameters - wind direction at releasing height at the beginning of experiment, temperature, wind speed at 10 or 40 m height and atmospheric boundary layer height from data files provided by [Irw12a], stability class was assessed by table 2.1 by cloudiness and wind speed near surface. The runs from 27.10.1977 and 02.08.1977 were omitted in evaluation and testing due to incompletion of their input parameters or concentration measurements, resulting in thirteen usable experiment runs.

4.6 Genetic Algorithm

Genetic algorithm optimizes the decision tree so that the calculated results come closer to the real-world measurements. In implementation, class *GeneticAlgorithm* deals with this task. Its main function *run* takes current decision tree, experiment runs used as training data and returns improved decision tree. Training data are in form of *map<string, vector<string>*>* where *string* is name of experiment's class (e.g. CopenhagenData) and *vector<string>** is vector with experiment's runs's dates that are to be used in evaluation. Part of the code of the method follows (allocation and deallocation of variables is left out).

```
DecisionTree* GeneticAlgorithm::run(DecisionTree *currentTree, map<string,
vector<string>*> trainingData)

{
    long generationCount=0;
    best = encodeIndividual(currentTree);
    while ((generationCount<maximumGenerations))
    {
        createPopulation(best);
        evaluatePopulation(trainingData);
        chooseBest();
        generationCount++;
    }
    DecisionTree* foundTree = decodeIndividual(best);
    return foundTree;
}
```

4.6.1 Encoding the Decision Tree and Decoding Its Chromosome

Chromosome corresponding to the decision tree is the vector of *double* values. For each node in preorder, values are stored in given order as shown in table 4.4. ID number is position of the node in preorder of the tree, root has ID 0. Split condition value is number assigned

to split condition in enumeration *UsabilityConditions*. For leaf nodes, split condition *nullCondition* with number 0 and split threshold -1 are there. Target model value is number assigned to specific model in enumeration *ImplementedSpecificModels*. For inner nodes, target model *nullModel* with number 0 and contribution ratio -1 are there. Bit can be changed has two values - 0 if node can not be changed, 1 if the inner node's split threshold can be changed and the leaf node's contribution ratio can be changed.

Decoding takes the chromosome and iteratively in preorder recreates the tree, as all the information about nodes are present in vector of *double* values.

Table 4.4: Part of chromosome in genetic algorithm representing decision tree node.

id	split condition	split threshold	target model	contribution ratio	can be changed

4.6.2 Creating the Population

Population is represented as array of chromosomes with fixed length during all run of algorithm. New generation of population is created from the best chromosome by its mutation.

The best individual is the individual with smallest cost value. If there is more than one such individual, then the one that is not the parent of current generation is selected as the parent of next one.

Algorithm offers three types of mutation that can be done on chromosome:

- adding new inner node - right before leaf node, so that the validity conditions are not threatened,
- changing split threshold on node that can be changed,
- changing contribution ratio on node that can be changed.

The third type has the highest default probability, as most of the training data available are suitable for Gaussian and Continuous Point Source Model, so their linear combination is the result of the system. Changed contribution ratios for these two models directly influence the coefficients of that linear combination.

4.6.3 Evaluating the Population

In population's evaluation every individual is evaluated and its cost value is stored in vector of *double* values. Individual's evaluation process iterates though all training data with following steps and average of normalized mean square errors is the cost value of the individual.

- method decodes the individual, i.e. creates the corresponding decision tree,
- *ModelInput* with the input parameters from training data is initialized,
- *ModelOutput* is initialized, with points identical with where measurements were taken,
- *ModelManager* calculates the *ModelOutput* with *ModelInput* parameters and decoded decision tree,
- calculated and measured concentrations are compared, normalized mean square error calculated.

Normalized mean square error (NMSE) is defined as

$$NMSE = \frac{mean((C_o - C_p)^2)}{mean(C_o) \cdot mean(C_p)}$$

where C_o are observed results, in our case the calculated concentrations, and C_p are predicted results, in our case the measured concentrations. The closer the error to 0, the better. Formally, NMSE represents the quadratic error of calculated concentrations in relation to the measured ones [WVMB05, p.2175].

4.6.4 Parameters of Genetic Algorithm

All parameters of genetic algorithm can be changed by user in training mode. They include:

- population size, values 1-100, small populations are recommended (5-15 individuals),
- maximum generations, values 1-1 000 000, values from 1 000 - 10 000 are recommended,
- mutation probabilities for three types of mutation, with the training data offered in implementation, only changing the contribution ratio is meaningful as explained in chapter 5.

The new changed decision tree returned by genetic algorithm replaces the old one and is saved in the file „tree.xml“ in current directory.

4.7 Graphical User Interface

Graphical user interface has been implemented in Qt. It consists of two parts - area of results's visualization and settings panel. The rendering of visualizations is done by class *QPainter*. Settings panel's contents depend on whether user or training mode is set in menu. In user mode, it includes forms for setting input parameters, selecting used specific models or the decision tree and visualization options. In training mode, list of experiments's data, their input parameters and forms for setting parameters of genetic algorithm are present. Visualization area is represented by class *VisualizationWidget* that overrides the method *paintEvent(QPaintEvent *event)*. This method paints over whole area the concentrations's values in calculated *ModelOutput*. The color ranges from green (0 or very small concentration) to red (highest concentration) and is set by logarithmic scale of concentration value. Beside visual representation of results, text output can be generated with exact values for all points. The user manual for interface is on CD attached to the thesis.

4.8 Modularity of the System

The system has been implemented as modular, with possibility to add new models of air pollution and real-world data. The detailed description of what needs to be done follows.

4.8.1 Adding a Specific Model to the System

When adding new specific model to the system, take these steps.

- implement the class as subclass of *Model*, override method *calculateConcentrations()* and store calculated values in *ModelOutput* in described way,

- add it to the enumeration *ImplementedSpecificModels* of the class *ModelsConditions*, include it in methods that convert the enumeration member to string and vice versa,
- specify how important is for the new model that conditions in *UsabilityConditions* are satisfied (1 - it is a validity conditions, 0.75 - recommended use of this model if this condition is true, 0.5 - does not matter, 0 - can not be satisfied) in method *initializeModelsConditionsValues()* of the class *ModelManager*,
- add case to method *getModelByName(string name)* of the class *DecisionTree*,
- add the model to the initial tree to the proper place in method *initializeDecisionTree()* of the class *DecisionTree*.

After adding new specific model to the system, it is recommended to rerun the genetic algorithm to find new decision tree that has learned from real-world data.

If also new split conditions are needed, do the following:

- add them to the enumeration *UsabilityConditions* of the class *ModelsConditions*, include them in methods that convert the enumeration member to string and vice versa,
- specify how important is for all implemented models that new conditions are satisfied (1 - it is a validity conditions, 0.75 - recommended use of this model if this condition is true, 0.5 - does not matter, 0 - can not be satisfied) in method *initializeModelsConditionsValues()* of the class *ModelManager*.

4.8.2 Adding an Experiment's Dataset to the System

When adding new dataset, this needs to be done.

- implement the class as subclass of *Data*, where every receptor's position is stored in the structure *Position* (its name and east-north coordinates needed), every measurement stored in the structure *Measurement* (coordinates of the receptor's position, average concentration in variable m),
- data are in format of *map<string, Measurement>* where string is the name of receptor's position,
- input is in format of class *ModelInput* described in section 4.2 ,
- vector *allDates* include all dates, or other names of different runs of the experiment, so that calling the method *getData(string date)* with any date from the vector will return data from it.

After adding new data to the system, genetic algorithm can use them to improve the decision tree, so rerun is recommended.

Chapter 5

Evaluation

The research question of this thesis are *How can we select and combine several specific models according to input parameters?* and *How can the system learn with real-world data?*. The hypothesis stated in section 2.5 says that the measured concentrations of pollutants will be similar to the concentrations calculated with our system, given the same input data or, at least, that the results of combined model will be closer to measured data than the results of the specific models themselves. The aim of the evaluation process is to prove this hypothesis.

5.1 Results with Specific Models

There are five specific models implemented in the system. The Gaussian model, Continuous point source model and approximation by method of lines can be used for calculation of the results with input parameters from Copenhagen and Cabauw experiment. The table 5.1 gives an average NMSE with these data. Also, error with initial decision tree is given for comparison. The NMSE is quite high, especially with Cabauw experiment's data. This causes the quite low performance of genetic algorithms in next section, as the results of the model combined by decision tree directly depend on the results of used specific models. In order to compare the results achieved by the system to the results of other (not implemented) models, genetic algorithm runs also on training data only from Copenhagen.

Table 5.1: NMSE between specific models and experiments's data.

model/data	Copenhagen	Cabauw	average
Gaussian model	0.282	2.367	1.325
ContinuousPointSourceModel	0.636	5.385	3.0105
combined model by initial decision tree	0.359	2.857	1.608

Numerical approximation by method of lines is time-demanding and experiment's measurements were taken in distance from source up to six kilometers. Therefore, this model is left out from genetic algorithm's evaluation and it is not recommended to calculate concentration in large distances. Instantaneous point and line source models have obviously *instantaneous source* validity condition. Therefore, they have not been evaluated on real-

world data. However, the screenshots from graphical user interface with an example input parameters suggests that the calculation gives quite correct results (see figures 5.1, 5.2).

Figure 5.1: Results of instantaneous point source model

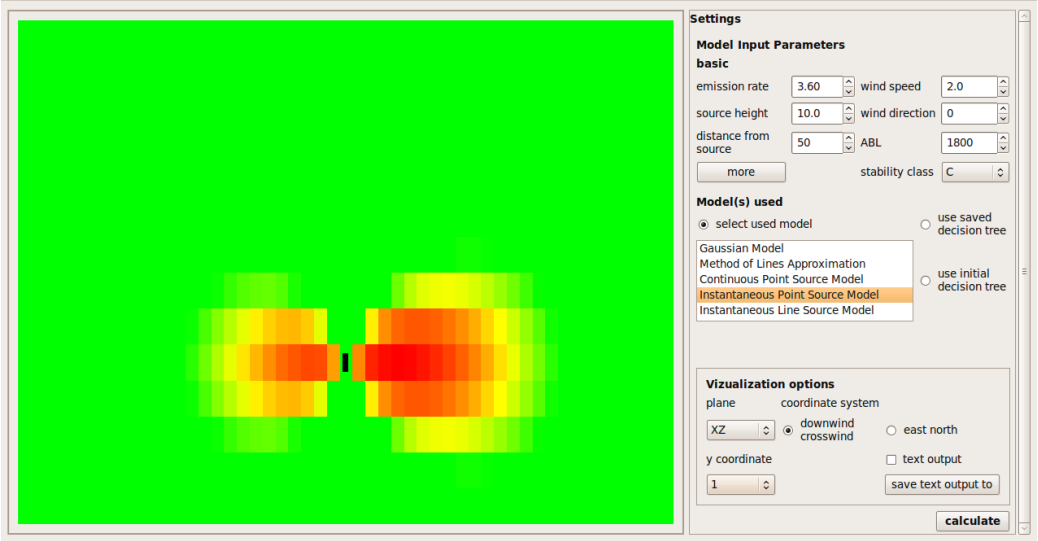
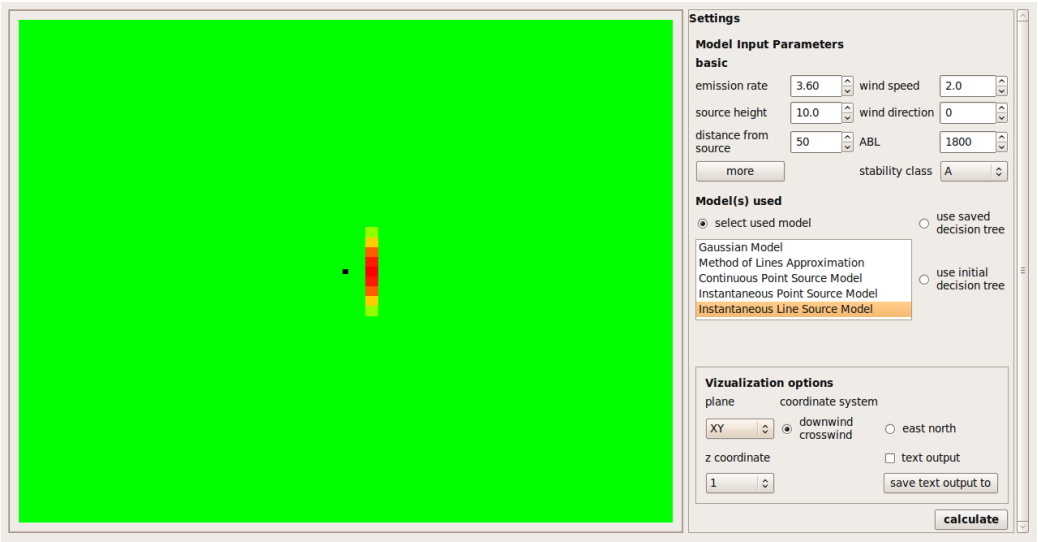


Figure 5.2: Results of instantaneous line source model



5.2 Results with Decision Tree Changed by Genetic Algorithm

The evaluation of effects that genetic algorithm has on the system's results is crucially significant for determining whether the whole *genetic algorithm changing the decision tree* process is meaningful. The process tries to find which values for genetic algorithm's param-

eters produce decision trees that have better NMSE between calculated and measured data than specific models themselves.

The procedure (one run of evaluation process) first randomly choose which runs from real-world datasets will be used as training data and which as evaluation data. Genetic algorithm then changes the decision tree so that it fits to the training data. After that, *ModelManager* calculates results with newly found decision tree, compares them to the evaluation data and computes the normalized mean square error. This value is referred as error of the run, error of the decision tree or NMSE in tables. The run is repeated with each configuration *(population size, maximum generations)* 10 times. Following parameters of genetic algorithm has been put into evaluation:

- the population sizes 5, 10, 15 and 20 individuals,
- maximum generations 10, 100, 1000, 5000, 10000 and 20000,
- mutation probabilities - as both datasets's input parameters satisfy the same conditions (point source, continuously emitting, almost constant wind speed and direction), the decision tree always selects Gaussian model and Continuous Point Source Model as appropriate to use. Therefore the only mutation operation is changing the contribution ratio and it has the probability set to 1.
- cost function - NMSE.

The evaluation process has been conducted twice. First, both dataset from Copenhagen and Cabauw experiment has been used as training and evaluation data. Secondly, only Copenhagen data has been used, so that the comparison with another (not implemented) specific models can be done.

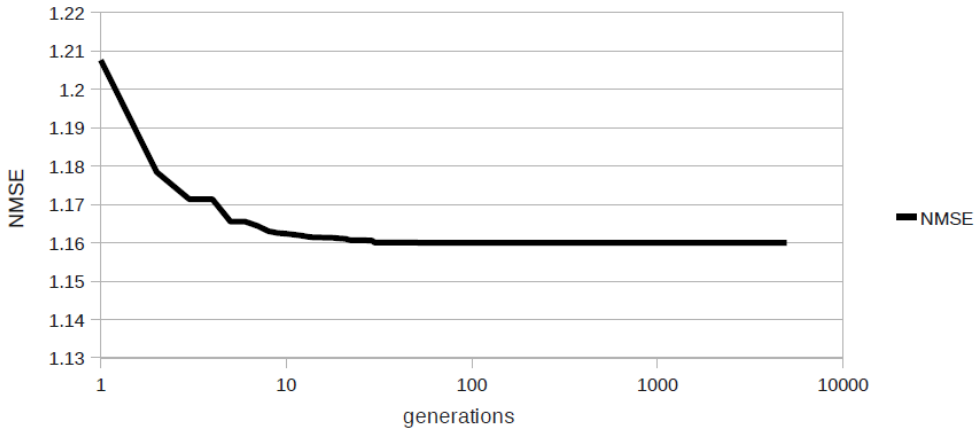
5.2.1 Copenhagen and Cabauw datasets used

In the first part of evaluation process datasets from both experiments were used. Training data for genetic algorithm includes (for each execution of genetic algorithm) random four runs from Copenhagen dataset and random eight runs from Cabauw dataset. Evaluation data consists of the remaining tree Copenhagen runs and five Cabauw runs. The results are a bit worse due to the fact that Cabauw data does not fit very well either Gaussian model, nor Continuous Point Source Model. However, the vast majority of algorithm's run results in decision tree whose combined model produces better results than specific models themselves.

As shown in graph 5.4, the errors in evaluation process do not follow the usual behaviour for maximum generations parameter. With genetic algorithms, the cost value is expected to decrease until the ideal value for given parameter in x axis is reached, after that it goes up. The graph shows that every population size the error fluctuates. The closest to the expected behaviour is population size 10. The error first goes down, it shows optimal error with 10000 generations and then goes up again (with the expectation of error with 5000 generations). That suggests that the maximum generations parameter's value does not determine overall results. The graph 5.3 shows one run of genetic algorithm in configuration (15, 5000). The error decreases mostly in a first few generations which explains why the number of maximum generations is not the determining factor of the result.

The parameter population size seems to have more influence, as shown in the graph 5.5. For maximum generations 10, 1000, 5000, 10000 the ideal population size is 10 or 15

Figure 5.3: The decrease of NMSE in one run of genetic algorithm with configuration (15, 5000).



individuals, for 100 maximum generations it is 20 individuals and for 20000 generations, 5 individuals are enough. Population size 10 and 15 individuals provide stable results with small standard deviation (only 0.049 for 15 and 0.052 for 10 individuals in population) over all maximum generations. Five and 20 individuals in population resulted in smallest observed average error (e.g. in (5, 20000) or (20, 100) configurations), or in highest observed average error (configurations (5, 1000) and (20, 10)). Additionally, configuration (5,30000) was run, but its average error 1.4 is far from relatively small average error of (5,20000). So, adding more generations does not improve results, even for small populations.

Figure 5.4: Average NMSE for all population sizes.

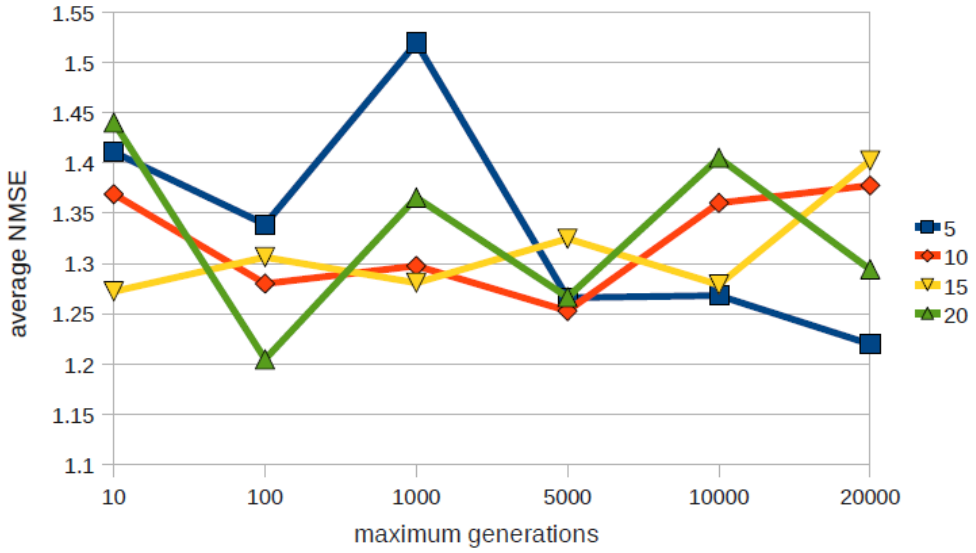
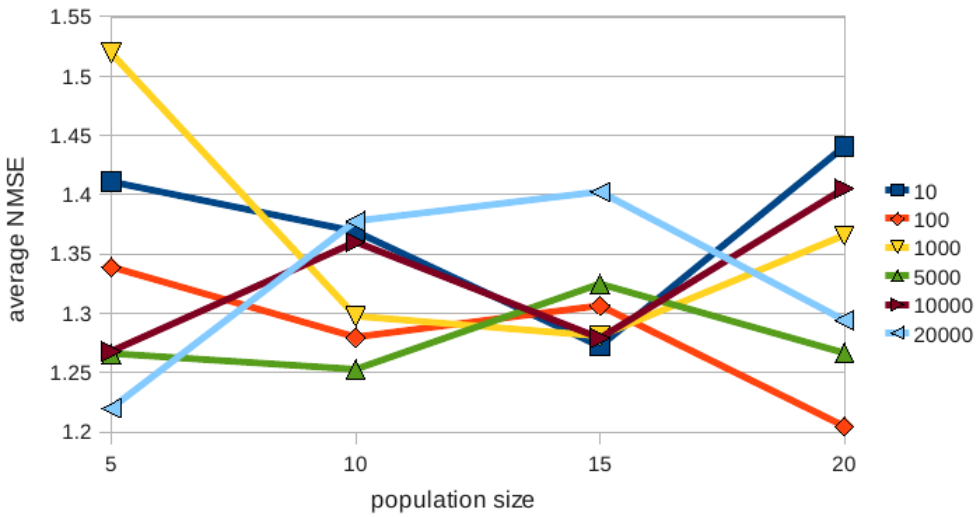


Figure 5.5: Average NMSE for all maximum generations.



Same graph, but for minimal NMSE calculated in 10 runs of evaluation procedure, confirms the importance of parameter population size and also the stability of values 10 and 15 for it (see graph 5.6). The standard deviation for minimal error with population size 15 is even smaller, just 0.02. Population size 5 again produces highly variable results. The smallest error of the first part of evaluation 0.739717 comes from configuration (5, 5000), but the highest minimal error comes from configuration (5, 1000). The contribution ratio for Gaussian model in the best tree is 1780.24, for Continuous Point Source Model 5.32052e-05. Maximal NMSE (see figure 5.7) again shows that the most straight line belongs to population size 15. The worst error found 1.92597 comes from configuration (10, 20000). So, the configurations (10/15, 1000-10000) has the highest probability of producing good enough results. With population size 15, the performance would vary only to minimal extent. However, the population size 5 has potential, especially with high maximum generations, but its results can differ very much from each other.

Tables with exact average, minimal and maximal NMSE are given in Appendix E.

Running Times

Running times of genetic algorithm directly correspond to the population size and maximum generations. Twice the size of population, twice the time - 190s for 5 individuals and 5000 generations, 380s for 10 individuals and 5000 generations. Twice the maximum generations, twice the time - 380s for 10 individuals and 5000 generations, 760s for 10 individuals and 5000 generations. The graph 5.8 proves that the relation between running time and these parameters is polynomial, not exponential. The slope of curve is converging when y axis is in logarithmic scale. If the relation is exponential, the curve would change to non-converging line. Another factor influencing the execution time is size of training data, in the second part of evaluation process with training data only from Copenhagen, the times were two times smaller. The average running times in seconds for all configurations are in table E.4 in appendix E.

Figure 5.6: Minimal NMSE for all maximum generations.

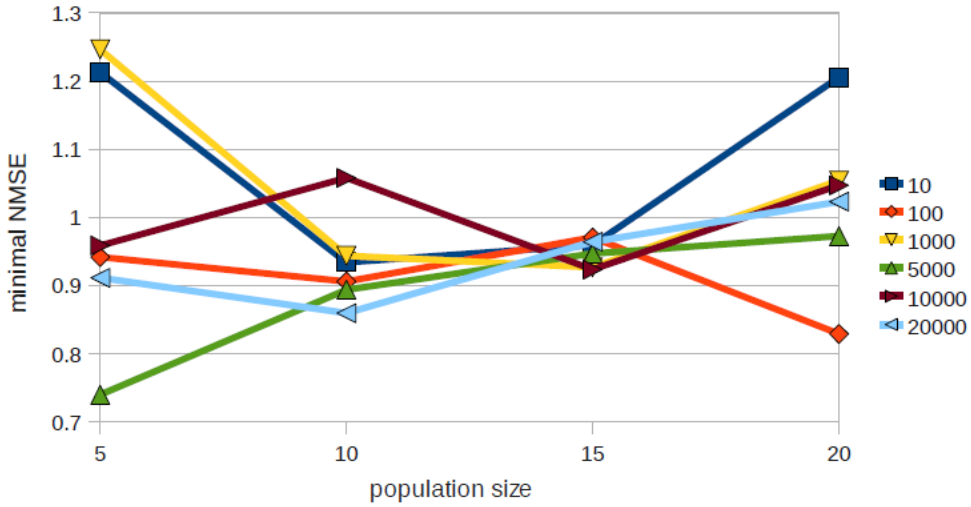
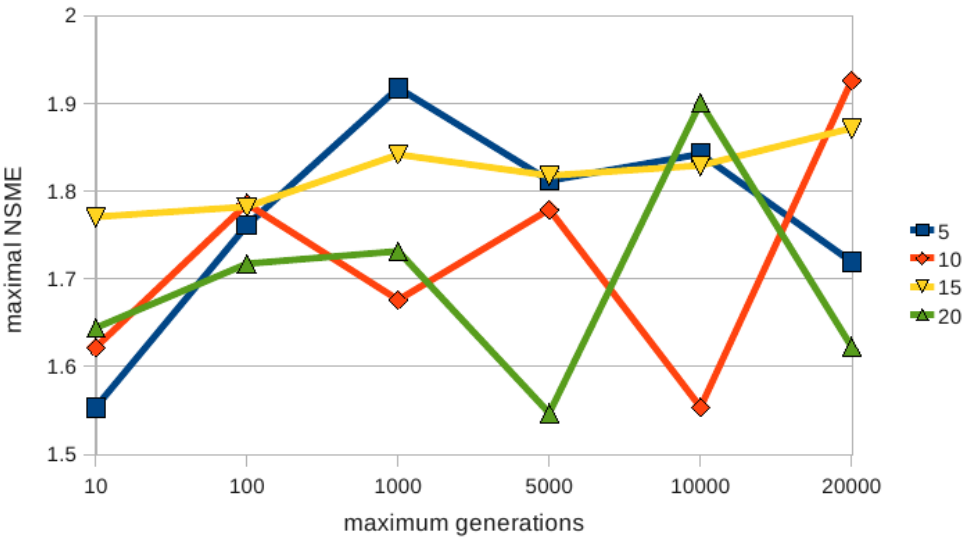


Figure 5.7: Maximal NMSE for all population sizes.



5.2.2 Copenhagen dataset used

Most papers focusing on air pollution modelling and simulation evaluate their systems only on Copenhagen data. Therefore, in the second part of evaluation process, only dataset from that experiment was used so that the system produce results comparable with other models. Training data are randomly chosen four run of Copenhagen experiment, evaluation data the other three runs. The behaviour is somewhat similar to the one described in previous section. Population sizes 10 and 15 give stable performance, smaller or bigger populations

Figure 5.8: Running times for all size populations.

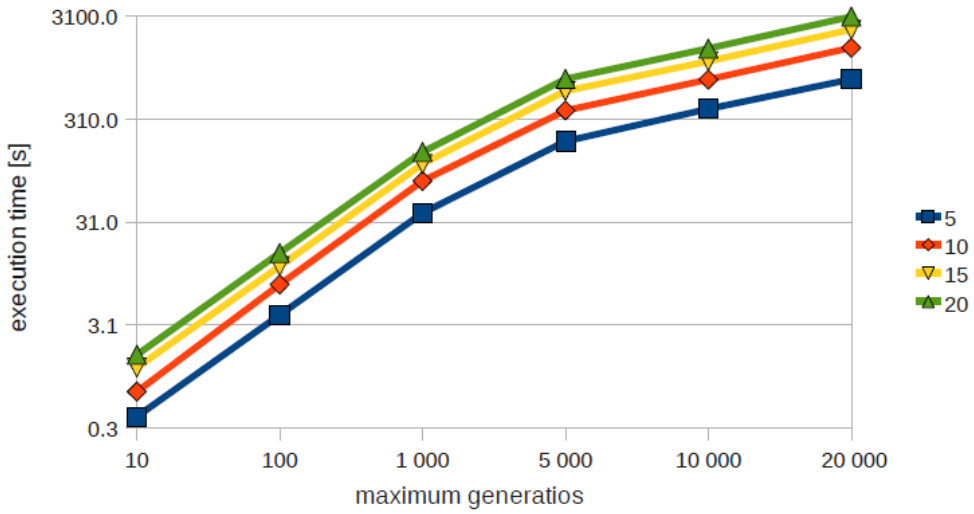


Table 5.2: Average NMSE with only Copenhagen dataset.

p/g	5	10	15	20
10	0.2958061	0.3013713	0.3093839	0.2848847
100	0.2639404	0.3125833	0.2814126	0.3119984
1000	0.2684008	0.2650861	0.3098842	0.2961745
5000	0.3160588	0.2591157	0.3147969	0.2376323
10000	0.2824322	0.2715296	0.3469239	0.313319
20000	0.2983143	0.2856961	0.3217635	0.2587044

give sometimes very good but sometimes quite bad results. However, population size 10 have smaller standard deviation of average and minimal results than 15 individuals. Computed average and minimal error are given in tables 5.2 and 5.3. Maximal error was around 0.4. In tables, p stands for population size and g for maximum generations.

Minimal error in this part of experiment is 0.12 found with three configurations - (5, 20000), (10, 10000) and (20, 5000), average ranged from 0.25 to 0.35. In the best tree of configuration (10,10000) the contribution ratio for Gaussian model is 2.82738, for Continuous point source model it is 0.04959. The table 5.4 gives an overview of other specific models. The system does not have the best results, but despite very simple specific models with quite high error (Gaussian model and Continuous Point Source Model), its results are comparable. The models of [Ulk00, WVMB05, MVT+05, MVBT06, CSR09] have highly specific application area, while this system can be used even with different input parameters.

5.3 Discussion

Based on collected results, the concentrations calculated by combined model built by decision tree changed by genetic algorithm gives, in general and mostly, are closer to real-world

Table 5.3: Minimal NMSE with only Copenhagen dataset.

p/g	5	10	15	20
10	0.210837	0.169711	0.180063	0.122315
100	0.16105	0.201617	0.161057	0.241657
1000	0.161076	0.161077	0.169873	0.222135
5000	0.231238	0.169873	0.222134	0.122312
10000	0.129188	0.122312	0.264106	0.276944
20000	0.122312	0.180036	0.161077	0.180036

Table 5.4: Comparison with other models.

model	NMSE with Copenhagen data
this system	0.12
used Gaussian model	0.3
used Continuous Point Source Model	0.636
[Ulk00] model GHIS	0.072
[Ulk00] model MM	0.072
[Ulk00] model MH	0.2
[WVMB05] [1]	0.06
[MVT+05]	0.07
[MVBT06]	0.09
[EE07]	0.08
[CSR09] double GITT model	0.04

measurements than concentrations calculated by specific models themselves. The hypothesis proposed in the analysis of this thesis has been proven to be true.

The evaluation has been affected by the lack of real-world data. Instantaneous models have not been properly validated as no experiments with instantaneous sources were found. With a lot of data with variable input parameters mutation operations changing inner nodes would be meaningful and probably effective. In this case, there is no point in adding new nodes into decision tree, when all training data satisfy same conditions.

The comparison between this system and other specific models shows that genetic algorithm is capable of producing the decision tree that gives rather good results despite the fact that only simple specific models are used. If other specific models, more complicated to implement but with better results, were part of decision tree, we can assume that the performance of the system would also improve.

Chapter 6

Conclusions

This master's thesis focuses on adaptive model for simulation of atmospheric pollution. The proposed solution is able to select appropriate specific models according to input parameters and combine them. Moreover, it is able to learn from real-world data and makes the selection and combination process more accurate. For these tasks, it uses methods of artificial intelligence - decision tree and genetic algorithm. The design of the system makes it rather easy to add new models or datasets to it. The system has been extensively evaluated on experiments from Copenhagen and Cabauw and the results prove that the combination of models gives better results than models themselves. Even with very simple analytical models, the results of the system are comparable with other (not implemented) models. Assumably, with more specific models implemented, the error decreases and the application area widens. With more experiments's datasets, the decision tree can learn and find better contribution ratios for more general model input parameters.

The future work on this project involve, of course, adding new specific models and datasets. Furthermore, chemical transformation of pollutants and terrain properties might be included in the calculation process. Also, combination of models based on dividing the area to few subareas where the results for each part are calculated with different, most suitable model as presented in design of the system has not been implemented. All these advanced features would result in even more general applicability and better performance of the system.

To conclude, the framework for selection and combination models of air pollution with ability to learn has been designed and implemented with promising results. The system shows very useful properties - the adaptiveness to the input parameters, the potential to improve with training and modularity.

Bibliography

[Ack02] Peter John Acklam. Lower tail quantile for standard normal distribution function, method in c language. http://home.online.no/~pjacklam/notes/invnorm/#Overview, 06 2002.

[BM04a] Marija Božnar and Primož Mlakar. Artificial neural network-based environmental models. In Sven-Erik Gryning and Francis Schiermeier, editors, *Air Pollution Modeling and Its Application XIV*, pages 483–492. Springer US, 2004.

[BM04b] Marija Božnar and Primož Mlakar. Use of neural networks in the field of air pollution modelling. In Carlos Borrego and Guy Schayes, editors, *Air Pollution Modeling and Its Application XV*, pages 375–383. Springer US, 2004.

[Bui01] Peter Builtjes. Major twentieth century milestones in air pollution modelling and its application. In Sven-Erik Gryning and Francis Schiermeier, editors, *Air Pollution Modeling and Its Application XIV*, pages 3–16. Springer US, 2001.

[CSR09] M. Cassol, S.Wortmann, and U. Rizza. Analytic modeling of two-dimensional transient atmospheric pollutant dispersion by double gitt and laplace transform techniques. *Environmental Modelling and Software*, 24:144–151, 2009.

[DPMH04] Andry A. Dudatiev, Yuliya Y. Podobna, Pavlo A. Molchanov, and Tatyana G. Holyeva. Fuzzy sets application for estimation of air polluted zone. In Carlos Borrego and Guy Schayes, editors, *Air Pollution Modeling and Its Application XV*, pages 311–318. Springer US, 2004.

[DvZ09] R. Dvořák, V. Šimek, and F. Zbořil. A numerical solution of the dispersion modeling in the planetary boundary layerche. *International Conference on Computational Intelligence, Modelling and Simulation, Brno*, pages 6–10, 2009.

[EE07] K.S.M. Essa and M. Embaby. New formulations of eddy diffusivity for solution of diffusion equation in a convective boundary layer. *Atmospheric Research*, 85:77–83, 2007.

[EHSZ08] M. El-Harbawi, M. Sa'ari, and A.R. Zulkifli. Air pollution modelling, simulation and computational methods: A review. In *Proceedings from ICERT 2008, International Conference on Environmental Research and Technology, Penang, Malaysia*, 2008.

[GLRD98] S.E. Gryning, E. Lyck, Roskilde. Wind Energy Risø National Lab., and AtmosphericPhysics Dept. *The Copenhagen tracer experiments: Reporting of measurements.* RISØ-R–1054. 1998.

[GMCT04] A.G. Goulart, D.M. Moreira, J.C. Carvalho, and T. Tirabassi. Derivation of eddy diffusivities from an unsteady turbulence spectrum. *Atmospheric Environment*, 38:6121–6124, 2004.

[HPM08] Sue Ellen Haupt, Antonello Pasini, and Caren Marzban. *Artificial Intelligence Methods in the Environmental Sciences.* Springer, 2008.

[Irw12a] John S. Irwin. Cabauw sf6 dispersion experiments 1977-1978 discussion. http://www.jsirwin.com/CabauwDiscussion.html, February 2012.

[Irw12b] John S. Irwin. Tracer data sets. http://www.jsirwin.com/Tracer_Data.html, February 2012.

[Isk10] Mohamed Iskandarani. Numerical methods - finite difference approximations. materials for msc321 scientific programming for atmospheric science. http://www.rsmas.miami.edu/personal/miskandarani/Courses/MSC321/, 2010.

[Jac05] M.Z. Jacobson. *Fundamentals of Atmospheric Modeling, 2nd edition.* Cambridge University Press, 2005.

[Kal06] Marcin Kalicinski. Rapidxml. http://rapidxml.sourceforge.net/, 2006.

[Kin09] Alexandra King. *CEE 6550 Mixing, Transport, and Transformation in the Environment, lecture notes.* Cornell University, 2009.

[Kuz10] Dmitri Kuzmin. Introduction to computational fluid dynamics, lecture notes. http://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/cfd.html, 2010.

[LH97] J. Lin and L.M. Hildemann. A generalized mathematical scheme to analytically solve the atmospheric diffusion equation with dry deposition. *Atmospheric Environment*, 31:59–71, 1997.

[MRRM00] M. Marín, V. Rauch, and A. Rojas-Molina. Cellular automata simulation of dispersion of pollutants. *Computational Materials Science*, 18:132–140, 2000.

[MVBT06] D.M. Moreira, M.T. Vilhena, D. Buske, and T. Tirabassi. The giltt solution of the advection diffusion equation for an inhomogeneous and nonstationary pbl. *Atmospheric Environment*, 40:3186–3194, 2006.

[MVT+05] D.M. Moreira, M.T. Vilhena, T. Tirabassi, D. Buske, and R. Cotta. Near-source atmospheric pollutant dispersion usingthe new giltt method. *Atmospheric Environment*, 39:6289–6294, 2005.

[NAD83] F.T.M. Nieuwstadt, R. Agterberg, and H. Van Duuren. Dispersion experiments with sulphur hexaflouride from the 213m high meteorological mast at cabauw in the netherlands. Technical report, Royal Netherlands Meteorological Institute, De Bilt (Netherlands), January 1983.

[SJ05] Scott A. Socolofsky and Gerhard H. Jirka. *CVEN 489-501: Special Topics in Mixing and Transport Processes in the Environment, lecture notes.* Texas A&M University, 5 edition, 2005. https://ceprofs.civil.tamu.edu/ssocolofsky/cven489/Book/Book.htm.

[Ulk00] A.G. Ulke. New turbulent parameterization for a dispersion model in the atmospheric boundary layer. *Atmospheric Environment*, 34:1029–1042, 2000.

[Val08] D. Vallero. *Fundamentals of Air Pollution, 4th edition.* Elsevier, 2008.

[WVMB05] S. Wortmann, M.T. Vilhena, D.M. Moreira, and D. Buske. A new analytical approach to simulate the pollutant dispersion in the pbl. *Atmospheric Environment*, 39:2171–2178, 2005.

[ZD07] Paolo Zannetti and Aaron Daly. An introduction to air pollution - definitions, classifications, and history. In P. Zannetti, D. Al-Ajmi, and S. Al-Rashied, editors, *Ambient Air Pollution*, chapter 1. The Arab School for Science and Technology, 2007.

Appendix A

Appendix Contents

- Contents of CD attached to the Thesis
- Extended Abstract written in Slovak language
- Used shortcuts
- Evaluation Details
- User Manual

Appendix B

Contents of CD

- /src - directory contains source code,
- /thesis - directory contains PDF version of this thesis,
- /guides/install - directory contains instalation guide,
- /guides/manual - directory contains user manual for graphical interface,
- /doc - directory contains technical documentation,
- /testing - directory contains evaluation details.

Appendix C

Extended Abstract in Slovak

Znečistené ovzdušie škodí životnému prostrediu aj ľudstvu. Na to, aby sme mohli riešiť existujúce a vyvarovať sa budúcich problémov, potrebujeme dobre rozumieť procesom, ktoré prebiehajú pri znečisťovaní ovzdušia. Počítačové modely využívajú rôzne matematické a štatistické princípy, aby našli vzťah medzi vstupnými parametrami a nameranými koncentráciami kontaminantu po danom čase v danej vzdialenosti od zdroja [Bui01]. Medzi vstupné parametre patria vlastnosti zdroja znečistenia (jeho pozícia, tvar, intenzita vypúšťania emisií, výstupná rýchlosť a teplota látky), počasie (predovšetkým rýchlosť a smer vetra, turbulencia vzduchu a okolitá teplota) a vlastnosti kontaminantu (jeho skupenstvo a chemická reaktivita) [Val08].

Súčasné modely používajú advekčno-difúzno-reakčnú rovnicu na výpočet očakávaných koncentrácií. K výsledkom sa dá dostať pomocou numerickej aproximácie, napríklad metódou konečných priamok alebo konečných objemov. Ďalším spôsobom je použiť niektoré zo známych analytických riešení, ktoré sú však použiteľné len za istých špecifických podmienok, ktoré zabezpečujú validitu riešenia. Prirodzene, tieto modely sú pomerne presné pri podmienkach, pre ktoré boli vyvinuté.

Model, ktorý by bol schopný prispôsobiť sa vstupným parametrom, vybrať vhodné špecifické modely, skombinovať ich do jedného systému a vypočítať výsledky, by mohol poskytnúť väčšiu perspektívu a širšie možnosti uplatnenia. Výber a skombinovanie modelov sú dve netriviálne úlohy a želanú adaptivitu systému by mohli zabezpečiť metódy umelej inteligencie. Navrhnuté riešenie využíva dve - genetický algoritmus a rozhodovací strom. Rozhodovací strom obsahuje informácie, podľa ktorých model vyberá a kombinuje špecifické modely podľa vstupných parametrov. Genetický algoritmus upravuje rozhodovací strom zmenou informácií, ktoré obsahuje a pridávaním nových tak, aby strom zohľadnil vlastnosti trénovacích dát. Toto riešenie je zrejme prvým pokusom vyberať a kombinovať modely podľa vstupu, avšak niekoľko systémov, ktoré spájajú dva modely dokopy boli vyvinuté [HPM08, ch.14].

Navrhnutý model pracuje z pohľadu používateľa ako akýkoľvek iný model. Užívateľ zadá vstupné parametre, systém vypočíta koncentrácie. Avšak, výpočet začína vybudovaním kombinovaného modelu pomocou rozhodovacieho stromu. Koncentrácie potom vypočíta kombinovaný model. Pri trénovaní systému sa po zadaní vstupných premenných a nameraných koncentrácií systém vypočíta výsledky podľa vstupu. Genetický algoritmus porovnáva namerané a vypočítané hodnoty a podľa toho upravuje rozhodovací strom.

Rozhodovací strom obsahuje všetky informácie týkajúce sa procesu výberu a kombinácie špecifických modelov. Koncové uzly obsahujú implementované špecifické modely a ku každému jeho *parameter presnosti*, ktorý zodpovedá tomu, ako presne daný model počíta

koncentrácie v porovnaní s reálnymi dátami. Vnútorné uzly reprezentujú podmienky ako *zdroj v tvare bodu* alebo *konštantná rýchlosť vetra*, ktoré sú buď podmienkami nutnými pre validitu modelov alebo určujú odporúčané použitie modelov za určitých podmienok, napríklad Langragov model je vhodný pre modelovanie dlhodobého časového rámca [Bui01, p.8]. Daný model sa vyberie vtedy, ak sú splnené všetky podmienky na ceste od koreňa stromu až k jeho koncovému uzlu. Pre každý model je tiež uložené v dátovej štruktúre, ktoré podmienky spĺňa a do akej miery (1.0 - podmienka musí byť splnená kvôli validite modelu, 0.75 - za tejto podmienky sa použitie modelu odporúča, 0 - splnenie tejto podmienky by spôsobilo nevaliditu modelu).

Výber a kombinácia modelov cez rozhodovací strom prebieha podobne ako klasifikácia rozhodovacím stromom. Postupne, od koreňa stromu, sa zisťuje, či vstupné parametre spĺňajú podmienky daného vnútorného uzlu. Ak je podmienka splnená, pokračuje sa doľava, ak nie, proces ide na pravý podstrom. Ak vnútorný uzol obsahuje podmienku *both*, proces pokračuje oboma podstromami a vracia sa zjednotenie nájdených koncových uzlov. V prípade viacerých špecifických modelov, ktoré rozhodovací strom určí za vhodné, sa celkový výsledok počíta ako lineárna kombinácia výsledkov jednotlivých modelov. Koeficienty sa počítajú na základe parametrov presnosti jednotlivých modelov.

Genetický algoritmus umožňuje adaptivitu systému tým, že optimalizuje rozhodovací strom. Vo všeobecnosti, genetický algoritmus iteratívne vyhodnocuje populáciu a vytvára novú pomocou operácii mutácie a kríženia na základe najlepších jedincov z predchádzajúcej. Rozhodovací strom, ako jedinec populácie, má podobu chromozómu, ktorá zjednodušuje operácie mutácie. Chromozóm stromu obsahuje identifikačné číslo uzlu a bitovú značku označujúcu či môže byť uzol menený, vnútorné uzly podmienku, koncové cieľový model a parameter presnosti pre každý uzol stromu v poradí preorder (rodič, ľavý potomok, pravý potomok). Populácie sú malé, nová populácia sa vytvára mutáciami najlepšieho jedinca. Používajú sa len operácie mutácie, konkrétne zmena parametru presnosti a pridanie nového uzlu. Pri vyhodnotení jedinca sa chromozóm naspäť dekóduje na rozhodovací strom, ktorý vyberie modely, vytvorí sa kombinovaný model, vypočítajú sa výsledky a porovnajú sa s nameranými hodnotami pri rovnakých vstupných parametroch. Vypočíta sa normalizovaná priemerná kvadratická chyba, ktorá je hodnotou ceny stromu.

Systém bol implementovaný v jazyku C++ a ohodnotený na dátach z experimentov Copenhagen [GLRD98] a Cabauw [NAD83]. Pri vyhodnotení systému sa potvrdilo, že kombinovaný model je schopný dosiahnuť lepšie výsledky (t.j. s menšou chybou) ako samotné modely. Dokonca aj napriek použitiu veľmi jednoduchých špecifických modelov s vysokými chybami systém našiel kombinácie, ktoré mali chybu porovnateľnú s inými (neimplementovanými) špecifickými modelmi.

Na záver, v tejto práci bol navrhnutý, implementovaný a vyhodnotený systém na výber a kombináciu špecifických modelov znečistenia ovzdušia so schopnosťou učiť sa, ktorý má sľubné výsledky. Systém vykazuje užitočné vlastnosti, predovšetkých prispôsobivosť vstupným podmienkam, potenciál zlepšovať sa trénovaním a modularitu.

Appendix D

Used shortcuts

In equations following symbols are used:

C concentration, $g.m^{-3}$

Q emission rate, $g.s^{-1}$

u wind speed, $m.s^{-1}$

u_x wind speed in constant x-direction, $m.s^{-1}$

E_x, E_y, E_z diffusion coeficients, $m^2.s^{-1}$

σ_y standard deviation of horizontal distribution, m

σ_z standard deviation of vertical distribution, m

L mixing height, m

h physical source height, m

H effective height of emission/plume rise height, m

x downwind distance, m

y crosswind distance, m

z receptor height from ground, m

k decay rate.

Appendix E

Evaluation Details

Tables E.1, E.2, E.3 show the error for all configurations (population size, maximum generations). The table E.4 shows execution times for all configurations on an *average machine* - Intel Core2 Duo CPU T6400 @ 2GHz and 3GB RAM. In tables, p stands for population size, g for maximum generations.

Table E.1: Average NMSE.

p/g	10	100	1 000	5 000	10 000	20 000
5	1.4108	1.33904	1.51946	1.26609	1.2681	1.22008
10	1.36905	1.28013	1.29773	1.25286	1.3603	1.37774
15	1.27229	1.30647	1.28092	1.32478	1.27913	1.40256
20	1.44049	1.20485	1.36574	1.26667	1.405176	1.294367

Table E.2: Minimal NMSE.

p/g	10	100	1 000	5 000	10 000	20 000
5	1.21264	0.941634	1.24631	0.739717	0.958695	0.911276
10	0.93402	0.9059	0.943628	0.893883	1.05788	0.859296
15	0.959287	0.970472	0.926727	0.94688	0.923268	0.964053
20	1.205	0.829031	1.05403	0.972529	1.04681	1.02294

Table E.3: Maximal NMSE.

p/g	10	100	1 000	5 000	10 000	20 000
5	1.55362	1.76112	1.91729	1.81167	1.84216	1.71926
10	1.62176	1.78553	1.67596	1.77901	1.55377	1.92597
15	1.77036	1.78195	1.84161	1.81739	1.82896	1.87138
20	1.64401	1.71706	1.73131	1.54686	1.90057	1.62219

Table E.4: Average running times of genetic algorithm in seconds.

p/g	10	100	1 000	5 000	10 000	20 000
5	<1	3.9	38.1	189.1	391.6	760.6
10	<1	7.7	77.9	377.2	753.3	1537.1
15	1.2	11.6	114.5	583.5	1137.4	2304
20	1.6	15.6	149.4	764.4	1508.8	3086.3